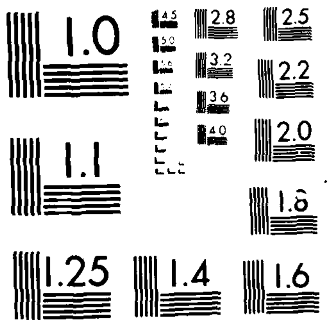
AD-A163 829 AUTOMATING KNOWLEDGE ACQUISITION IN A FLIGHTLINE ROBOT 1/2
(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH
SCHOOL OF ENGINEERING W M CLIFFORD DEC 85
UNCLASSIFIED AFIT/GE/ENG/85D-7 F/G 6/4 NL



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS 1963-A

(1)

AUTOMATING KNOWLEDGE ACQUISITION IN A FLIGHTLINE ROBOT

THESIS

William M. Clifford
Captain, USAF

AFIT/GE/ENG/85D-7

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GE/ENG/85D-7

AUTOMATING KNOWLEDGE ACQUISITION IN A FLIGHTLINE ROBOT

THESIS

William M. Clifford
Captain, USAF

AFIT/GE/ENG/85D-7

AFIT/GE/ENG/85D-7

AUTOMATING KNOWLEDGE ACQUISITION IN A FLIGHTLINE ROBOT

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

William M. Clifford. B.S.

Captain, USAF

December 1985

Approved for public release; distribution unlimited

Preface

Hardware has developed to the point that a mobile robot performing tasks on the flightline is within our grasp; however, software to drive it has not. The knowledge needed to perform simple jobs can be staggering. The best source of this knowledge is the domain expert for the particular task. Most domain experts would require assistance in transfering the knowledge to any system that exists today. A truly intelligent robot would assist its trainer in accomplishing this transference of knowledge.

I thank the many who have supported me. The Air Force and AFIT provided the opportunity to pursue my desire to know why. Steve Cross, my advisor, demonstrated an insight into both engineering and the mind. Tim Anderson, my co-advisor, suggested directions to turn when blinders kept them from sight. Matthew Kabrisky, my reader, convinced me to study at AFIT and then taught me when I wasn't looking. He provided a model for a lifetime and I would depart happy having half filled his shoes.

To my wife, Linda, I owe the greatest debt. Her understanding and compassion deserve far more than just acknowlegment. This is a debt that I will enjoy repaying for the rest of my years.

William M. Clifford

Table of Contents

List of Figures

List of Tables

Abstract

Robots enjoy widespread use in industry on simplistic repetitive tasks in a controlled environment. Tasks in the military domain, particularly the flightline, require mobility and intelligence. While the mobility issue is being addressed, the intelligence issue is not. By giving the robot the ability to learn from a novice, the robot could be placed on the flightline and learn what it needs to know from the domain experts.

Rather than deal with a toy problem, this work takes the actual refueling "Job Guides" for two aircraft and show how these can be used directly by a computer. The process involves three steps. First the text is transformed into a common natural language processor form. Second the forms are expanded by applying expert system and planning techniques to include missing domain and world knowledge. Finally the forms are examined to allow the commonalty between the two refueling tasks to be extracted.

Discussion includes background on learning, natural language processing using Conceptual Dependency representation, and planning using the Stanford Research Institute Problem Solver (STRIPS).

Introduction

Background

The Air Force is currently investigating the feasibility of using a mobile robot on the flightline. "A mobile autonomous robot capable of performing simple aircraft maintenance tasks could protect many lives from NBC (nuclear, biological, chemical) exposure while allowing the Air Force to maintain its combat readiness" (8:I-1). In addition to the protection offered against NBC exposure, as research and technology advance, other hazardous and mundane tasks might also be delegated to such robots. Masters candidates at the Air Force Institute of Technology (AFIT) have investigated various aspects of a flightline robot (8,28,33,39). Within AFIT, knowledge acquisition in such a robot has previously gone unaddressed.

Experts in the field of Artificial Intelligence (AI) state that a large knowledge base is required to make systems perform intelligently (10:6). Further, such knowledge bases cannot be static and still serve well in an environment as dynamic as a flightline. Programming robots is currently done either by leading them through the motions desired or by programming the motions in VAL or some other specialized robot programming language (20:12-58 to 12-68). Both methods are slow, and labor intensive, and require special training on the part of the programmer. While any method that would make this task easier will prove useful in the short term, transfering the burden of acquiring the knowledge to the robot and removing the expert trainer from the loop will prove much more useful in the long term.

Ideally, robots would be programmed in ways similar to the way people are taught. Research in this area is being done at The University of Connecticut as well as other universities (27:12-79 to 12-83). By understanding the difficulties in accomplishing this, the Air Force will make better decisions in the areas of acquisition and research funding. Further, in complex systems, software development generally takes as long as hardware development. It is important that the software be investigated now to allow the system to be deployed as soon as technology permits.

Problem

Students at AFIT have addressed various aspects of a flightline robot. Taylor [39] did an analysis of some of the tasks to be performed. Owen [33] investigated using a laser barcode scanner to identify position. Clifford and Schneider [8] improved upon the Hero-1 robot to add sonar and an optical shaft encoder. Monaghan [28] developed a navigation routine in Prolog. Other hardware improvements are in progress. This research investigates knowledge acquisition in a flightline robot.

The task addressed in this study is teaching a robot the general concept of refueling aircraft by providing it with instances of refueling specific aircraft. The research task is to examine how to present these refueling instances to the robot and to examine the suitability of one learning approach in the flightline domain. To enable this to be accomplished, the flightline domain is specified more precisely than has been attempted in the past. With the domain defined, the research concentrates on the study of the transference of knowledge. Particularly, it addresses how knowledge might be transformed, internalized, and reordered to make it useful. Since refueling is a task presented as numerous ill-defined and incomplete sequential subtasks, the problem of planning is also addressed.

Scope

As AFIT's first look at learning, specifically learning in the flightline robot domain, the breath of the material covered is initially very large. Background chapters provide the foundation necessary to consider the subject.

This research includes the design of an overall operating environment for flightline robots. By looking closely at how robots might be utilized, a basis for making engineering decisions is available. The concept of the flightline robot is refined sufficiently so that specific tasks can be discussed.

The theory of learning is covered in enough depth to allow the reader to understand the subclass of learning that is pursued. Since the input to the system is in the form of written text, natural language understanding is defined and the Schank's work on Conceptual Dependencies (38) is introduced. Finally, a connection is made between generalization and planning in this domain and a description of the Stanford Research Institute Problem Solver (STRIPS) as a planning tool for robots (18) is included. While planning may be involved in learning in a number of ways, the discussion here centers on the need to complete partial plans to allow them to be compared.

With the foundation established, three aspects of knowledge acquisition are examined. The first is the transformation of the knowledge as it exists in the environment (the checklists, or more properly, the Job Guides) to a operationalized form that a computer can more readily operate on. The second is to fill missing details with a planning phase. It is common to find that the guidance given by the Job Guide is missing steps that the authors of the Job Guide took for granted. A more detailed discussion of operationalization and planning is presented later. The third aspect of knowledge acquisition investigated is the reordering phase. This can be thought of as a data reduction phase and is accomplished by attempting to generalize specific instances into an overall concept. The rational for selection of the method used is also presented.

Assumptions

Since learning in humans is often dependent on verbal communication and since a large subset of humans is very accustomed to communicating in English, some method for the machine to interpret English language will be necessary. It is assumed that a machine can interpret spoken English and convert it into a representation that is convenient for the machine to operate on. It is further assumed that there is nothing fundamentally different between this domain and others that have successfully demonstrated natural language understanding [38]. Finally it is assumed that with sufficient enhancements to a standard representation scheme, adequacy of representation will not be a problem.

Some form of representing the world model is required. It is assumed, based on some success in the literature in similar domains [18], that first-order predicate calculus well formed formulas (wffs) are an adequate form to represent the current world state and goal state in the flightline domain.

It is presumed reasonable that some set of primative operations can be provided to the robot and that these operations can be designed with sufficient generality that they can be used to serve a number of goals. It appears that currently no one knows how to make a computer adapt an operator to do something for which the original designer did not provide.

Delays in executing the procedures due to processing will be ignored. It it assumed that these can be made to operate in "real-time" through optimization of algorithms, off-line processing, and through advances in hardware. Imagining other ways to solve the problem of computational complexity is left to the reader.

The aspect of learning termed the "Performance Element" (9:327) is not included in this research. It is assumed that initially a robot would be supervised and that the supervisor would provide the type of feedback that this function provides.

Finally, it is assumed that the trainer is a benevolent creature and will not mislead the robot. This closed world assumption is necessary to make the problem tractable. Determining if an algorithm developed under such an environment could be extended to work in the real world is not addressed.

Summary of Current Knowledge

Learning can be partitioned into skill refinement and knowledge acquisition. Skill refinement, commonly referred to as practice, appears to be required in humans. Adaptive control theory is the discipline that addresses this kind of learning in machines (6:6) and will not be addressed in this research project.

Knowledge acquisition is obtaining, reformating, and storing information that allows one to do a new job or to do an old job in a new way (9:326). Knowledge acquisition can be divided into rote learning, learning by being told, learning from examples, and learning by analogy (9:328). Programming can be considered an instance of rote learning. Learning by being told, also called advice taking or advice initiated task accomplishment, has been accomplished by computers with limited success (4:87-105,9:350-359). Learning from examples, also called induction and generalization, has also been demonstrated with limited success (41:385-408). Learning by analogy has not been accomplished though some argue that this is the only method by which man learns and in time this could prove the most useful form of learning (34).

A process that is closely related to learning is planning. This activity has been addressed in a number of research efforts and the STRIPS planner is commonly singled out for study. STRIPS uses a predicate calculus wff to describe a world model, operators that indicate how that world model may be acted upon, and a wff that represents a goal to be made true in the current world model. STRIPS follows a means-ends analysis approach to reduce the search of available operators to those that offer the possibility of leading to a solution.

Approach and Equipment

The Job Guide for fueling the F-15 [13:27-2 to 27-15] and for fueling the F-16 [14:2-1 to 2-27] will be translated into a complete and consistent internal representation of meaning that is amenable to further processing. Conceptual Dependency (CD) form [38], a well established paragon was chosen as the approach. This form is then expanded upon to operationalize it; that is, to provide details the Job Guide authors assumed were understood. These operationalized CDs are examined to select a predicate calculus wff that best describes the task to be accomplished. These wffs are used to search a data base of STRIPS-like operators to determine which operation is called for. These STRIPS-like operators then form the basis of a script (list of operations) that represents refueling of an aircraft. The two scripts generated in this manner are compared using a generalization operator. The result of this operation forms the basis for a generalized refueling plan.

To demonstrate proof of concept, a system consisting of a transformation (natural language understander and operationalizer) step, a planning step, and a generalizer step is constructed. Some operations that would require excessive time and reproduce results already obtained by other research are performed by hand to allow the other operations to be investigated. All programming is done in Zeta-Lisp(tm) on a Symbolics 3670(tm) computer.

Overview

Chapter 2 represents a detailed analysis of the robot on the flightline domain and knowledge acquisition in that domain. Chapter 3 presents the theory of learning with examples in the flightline domain. It introduces natural language understanding and also includes a discussion on planning. Chapter 4 describes the system that would allow spoken input to be internalized by the robot. Chapter 5 covers the implementation of the system. Chapter 6 analyzes the results and Chapter 7 suggests directions for further study.

Detailed Analysis

Introduction

Past AFIT thesis efforts have identified the need for an "android" (39) or "mobile autonomous robot"(8,28,33) to accomplish tasks on the flightline. While one effort mentioned the need for the robot to "learn and/or be programmed" (28:2) none have examined how high level training might occur. To be a truly useful system, the flightline robot will need to exploit some form of automated knowledge acquisition. This chapter will examine the concept of a robot on the flightline. It will then discuss automated knowledge acquisition and finally it will discuss constraints that would apply to such a system.

The Flightline Environment

To provide the essential flavor of a flightline, it consists of large areas of relatively flat pavement to park aircraft. The bounds of the flightline are generally service roads, taxiways, hangars and grassy areas. The surface can be either concrete or asphalt painted with lines to denote aircraft taxi paths (a single line) and ground vehicle virtual roads (two boundaries and a center line). Additional lines may exist that are no longer used and lines may exist to denote security areas. Positioned across the surface are depressions containing metal grounding points.

Additionally depressions may exist that contain tie down points. The final surface feature that may exist is the fuel pit. These pits are capped fittings that can serve as a source of fuel by connecting a pump to the fitting.

Obstacles on the surface consist of aircraft, people, aircrew transportation buses, fuel trucks, maintenance trucks, maintenance stands, power carts, fire extinguishers, and occasionally fire trucks. Litter is controlled to prevent damage to jet engines. For security reasons, the number of objects is small and controlled and moving traffic travels slowly (15 mph). The flightline is predominantly a closed environment and for this reason navigation by a mobile robot should not prove too difficult. One task that a robot might help perform is the refueling task.

The Refueling Task

Refueling an aircraft parallels refueling a car but additional safety precautions are taken that make the task more complex. The number of people required varies from aircraft to aircraft. For the F-15 and F-16 three people are required. One person stationed off the nose of the aircraft supervises the refueling operation, scans the aircraft for obvious leaks, and observes for hazards, particularly sources of ignition. A second person positions the fire extinguisher upwind, ensures the aircraft is still grounded, grounds the fuel truck to the aircraft and the fuel truck to the earth, and operates the fuel hose and nozzle. The third person operates the fuel truck. If the aircraft electrical power is used during refueling (not common in fighter type aircraft) a fourth man is required to be in the cockpit. All these people need to synchronize their activities and thus need to have knowledge of all of the tasks. Appendix A contains the F-15 and F-16 refueling Job Guides side by side with conceptually equivalent steps aligned. With the domain and the task outlined, the use of robots can be addressed.

The Need for a Mobile Autonomous Robot

Why Robots? The automobile industry is heavily involved in the use of robotics in manufacturing. The primary reasons they use robots are safety and cost effectiveness. Freeing labor from menial tasks has also been given as a justification. It is the author's belief that this last justification does not actually affect whether or not robotics are applied and only safety and cost effectiveness will be addressed.

Previous AFIT thesis have addressed the safety aspect by describing a nuclear, biological, chemical (NBC) contaminated environment in which the Air Force might be forced to operate (28:1). While NBC contamination may have been sufficient premise for a thesis, other important aspects have gone unmentioned. Every year airmen are killed performing necessary tasks on aircraft. Servicing liquid oxygen and hydrozine systems on today's aircraft are hazardous tasks. The Air Force pays hazardous duty pay to airmen doing some of these tasks. And every year lives are lost in related accidents. Other simple maintenance tasks also take lives. Through the use of robotics, some of these lives could be saved.

Cost effectiveness of robots in the flightline domain should also be considered. Given that hardware costs will continue to drop and that labor costs will continue to rise, robots will replace people in many positions. This replacement could be a complete elimination of people for some tasks and reducing the number of people required to perform others. The use does not and should not be restricted to wartime. Most of the DoD budget is spent during peacetime conditions. Further, this expense is considered a cost of maintaining readiness. Robots should be used in peacetime for their cost advantages and to ensure their job will be accomplished successfully in a wartime environment. If the Air Force is not ready to apply the technology when it becomes available, money will be wasted doing things the old way.

Why Mobile? While industry is applying robotics to tasks every day, the Air Force is hampered in that many of the tasks it would like to automate are unique (22). The aircraft are never quite parked in the same exact location.

Even with the level of standardization stressed, not every aircraft will have the same markings in the same place. Some have suggested that the flightline domain could be redesigned for the robot. "To facilitate android maintenance, some additional modifications to the aircraft and its components, as well as to flight line facilities, will probably need to be done" (39:18). This is a reasonable comment but must not be taken to mean that a properly designed flightline would eliminate most of the needs for mobility. Some may envision automated refueling pits and other improvements. What this position fails to recognize is the mobile nature of the military. A mobile robot could be positioned at a captured field, as in the Grenada expedition, and with some adaptive training could perform its tasks. Thus, deployable robotics applied to the flightline will prove more powerful.

There are additional implications when considering the modification of facilities. Costs involved in modifying buildings will seriously shift the point of cost effectiveness for robots. The time involved will also be a prime consideration. The effects of such changes on deployablity will make the robot useless away from home station. Modifying aircraft, taking both the cost and time considerations, shifts the robot at least a decade further into the future. A mobile robot that can work in the current environment with an absolute minimum of change to that environment is clearly the superior direction to proceed.

Why Semi-Autonomous? A discussion of robot autonomy can result in an argument in semantics. The dictionary defines autonomous as "self-governing, spontaneous, or independent" (Random House). This is not any commander's idea of an ideal troop. A commander is looking for an obedient individual. It is the synergism of effort that makes a unit powerful. The idea of a robot who roams the flightline looking for things to do is absurd. At the other extreme, a robot which is simply a teleoperated device fails to achieve the force multiplicity that is possible.

The ideal robot would be capable of handling complex tasks without additional input until the task is complete or an impasse is reached. This robot would also have a communications link to commmand and control so that status could be checked and to suspend or cancel tasks to allow redirection of resources.

This centralized control could handle more than one robot and if a robot is an economical tool it becomes reasonable to discuss applying a number of robots to a number of tasks. An additional feature of centralized control is that it would allow sharing of resources, especially knowledge, among a number of robots. When a central plan processor obtains new information through one robot or some other source, all robots can benefit from the new information. By having some autonomous capabilities within the robots, the centralized processor would not become saturated as easily. (Redundancy in centralized control would provide protection from single point failure.)

In a Stanford report the term "semi-autonomous" is used to describe a robot with some autonomous features. The domain addressed in the report is the surface of Mars. Because of the communications time delay to a robot on Mars it is not reasonable to operate by making a small input and waiting 30 minutes to see the result before another move can be made. Some autonomous capabilities are necessary. It should be easy to see that most robots in most domains would be more useful if they did not need constant outside input to perform their tasks (29:3). Thus, there is precedent for terming this device semi-autonomous.

The Need for Automated Knowledge Acquisition

Why Knowledge? One credo of Artificial Intelligence (AI) is "in the knowledge lies the power" (10:8). When a crew chief fuels an aircraft he makes use of knowledge he has acquired over his lifetime. If someone tells him that smoking is not allowed within fifty feet of refueling operations, he has the knowledge to figure out that the likely concern is the flammability of fuel and has nothing to do with, say, the alleged cancerous properties of smoking. When he encounters the filler cap, he can rely on similarities between this cap and an automobile fuel cap to hypothesize how the cap might be removed. Domain and world knowledge constantly come into play.

Why Acquisition? While it may be reasonable to program concepts of Physics, Chemistry, and many of the "fixed" pieces of knowledge, most of what one knows needs to be added to or altered as new tasks are encountered. Consider the same crew chief who now encounters a fuel cap that does not come off by simply twisting. He might recognize the existence of a small lever and experimentally determine that the lever needs to be lifted, freeing the fuel cap so that it may be twisted off. He will probably remember this information so that next time he encounters the same type of fuel cap, he can remove it directly. He is continuously ready to both depend on knowledge he has and yet alter it when it fails.

Acquisition is necessary for other reasons. No two flightlines are alike and moving from one location to another will require rebuilding the knowledge base. Even if only one flightline is considered, changes to that flightline occur over time; potholes develop, vehicles move.

Why Automated? Since the robot is going to have to acquire knowledge, the mechanism needs to be addressed. Currently robots are trained primarily by moving them through the path that is desired. For a robot on the flightline it might take months to step through all the tasks one would like to be able to accomplish. Even if this could be done, a robot trained this way would not be able to handle obstacles or other changes in the environment. By programming subtasks and assembling subtasks into tasks, the time might be reduced but this implies a knowledgeable programmer to handle the training. This is not acceptable. It is cost prohibitive, but more importantly a knowledgeable programmer may not always be available in a combat zone. Finally, by automating the task, human factors can be better addressed. One operator may not be able to operate the robot with joysticks and perform the task. Another operator may not be able to fully describe in words how the task is to be accomplished. Characteristics of each task and each teacher are factors in deciding how to transfer the knowledge best.

Constraints

Many constraints must be satisfied in a domain as complex as a flightline. These may be labeled as limited by technology, or limited by nature. The clever reader will realize that the boundaries of these categories are fuzzy and change with time. Acknowledging this and purely for convenience, the distinction will be made in the following discussion. A discussion on safety considerations follows separately because it does not fit in either category and because its importance rates special consideration.

Limited by Technology. Speech recognition, and the closely related area of natural language understanding, appear to be coming as processor power and specialized hardware become available. Such an interface to a knowledge acquisition system will speed up knowledge transfer by orders of magnitude. But with this interface comes the vagaries of English. Still needed to be addressed is how to avoid misunderstanding in a system that depends on English as its input language. Consider the sentence: "Ground the aircraft." To the crew chief that has just watched the aircraft taxi in this probably means "Electrically connect the airframe to the earth." To the pilot who has experienced uncommanded flight control inputs, this probably means "Don't let this aircraft fly until you find the source of the problem." Man is able to handle these differences using context and voice inflections. He also uses backtracking. When an interpretation he made fails to fit the discussion, he can think back to find a different interpretation. A machine should be able to do the same thing. Still, excessive amounts of backtracking will make the conversation hard to follow. The state of the art with machines has a long way to go.

Other problems that are traditionally labeled AI abound in learning. A knowledge representation scheme that lends itself to extensive reorganization has not been demonstrated. A revised STRIPS [11:226-268] attempted to turn selective pieces of constant information into variables when it is apparently necessary to do so. This included introducing outside information for specific cases and did not solve the problem for the general case. Turning variables into constants is an unsolved problem. That is not to say that these tasks are impossible. The very fact that man does it proves that it is possible. But a representation that can be applied to a computer has not yet been discovered.

Another learning related AI problem is one of search. Problems that require an exhaustive search of a large solution space will not be solved simply because processor speeds increase. Knowledge must be applied to reduce the search space to one that is manageable. The way in which domain knowledge in included has not been formalized but is done on a case by case basis. The next step may be to program a computer to make hypothesis that may reduce the search space or may lead down a garden path. Man seems to use something like this to successfully solve problems that should exceed his computational capacity. The final word on the search problem is not in.

Limited by Nature. Bandwidth limitations will appear throughout the system. In a traditional sense, the communication channel to command and control will be bandwidth limited. In a multi-robot environment that attempts to use an already saturated military RF spectrum, some limitations will have to be placed on information that is passed. A system that is designed to process vision information remotely will have to be rethought; unencoded video requires a high bandwidth. In a less traditional sense, for the robot to function in an NBC environment, it will have to deal with restricted voice communication when its operator is wearing masks, breathing devices, and protective clothing.

Noise is an ever present engineering limitation. In the learning environment this could be anything from mildly distracting superfluous inputs to inputs that are labeled true which are actually false. Until learning is explored more fully one cannot even guess at all of the ways noise will effect the system. One can hypothesize that man's immunity to noise may be directly related to the slowness with which he learns. If this is true, tradeoffs will have to be made. Few organizations will tolerate a twenty year training period before useful work is accomplished.

A previous thesis [39] presented other limitations and considerations. It would be redundant to cover them here. That thesis described a single android working on the flightline. It is left as an exercise to the reader to consult that thesis and imagine the situation of multiple robots in the flightline domain. One important omission has been the consideration of safety issues.

Safety. No engineering effort is complete without addressing safety implications of the project. Robotic issues require the same consideration but suffer from a complexity that silently encourages the designer to not mention them. This section is included to bring safety considerations out into the open where they belong.

To indicate the magnitude of the problem, consider the time span between the introduction of the automobile and its rise to the dubious position as the number one non-natural cause of death in the United States. All wars combined have killed about 575,000 U. S. citizens (21:419-420) while automobile accidents kill about 50,000 U. S. citizens per year (21:777). Had the issue of auto safety been more carefully considered, the design of the automobile may have proceeded in a different direction. The magnitude of the difference between automobile safety and robotic safety may not be that great.

The economic advantage of replacing manual labor with robotics makes rapid development very attractive. Omni magazine relates an Upjohn Institute for Employment Research study that puts the number of robotic devices in the 1990s at 50,000, a ten fold increase in ten years (19:97). If expansion continues at this rate, and past experiences with technological developments indicates that it should, the robot will become as common an item as the family auto.

Like the auto, the robot becomes a deadly weapon when mishandled. If no precautions are taken at all, the death rate for robot related accidents will skyrocket. That it might approach the death rate of the automobile is not easy to accept, but few would have accepted a similar prediction in 1920 for the horseless carriage.

Isaac Asimov addressed this problem with his now famous three rules of robotics:

> #1 A robot may not injure a human being, or, through inaction, allow a human being to come to harm.
>
> #2 A robot must obey orders given it by human beings except where such orders would conflict with the First Law.
>
> #3 A robot must protect its own existence as long as such protection does not conflict with the First or Second Law (1:1).

At first glance the list looks complete but Asimov notes that there are glitches in it. For one, a robot could do nothing until it could determine all future effects any action it might take. It could only be certain after predicting all future events; the average task would take an eternity to complete. For another, a robot could find itself in a situation where action would result in the death of one person while inaction would result in the death of another.

It is not the author's purpose to present a final solution. Instead, the reader needs to be aware that the situation is a significant one. To support this, the same

OMNI magazine article (19) relates the first documented case of a U. S. citizen being killed by a robot. On 25 January 1979, Robert Williams, while working at Ford's Michigan Casting Center in Flat Rock Michigan was crushed by a 2,500 pound robot that had no capability to detect humans and no reliable method to warn of its presence or intended actions. While no amount of effort will bring Mr. Williams back, the events that lead up to his death and the $10 million finding against the Unit Handling Systems of Litton Industries should be required reading for anyone considering robotic equipment.

The military stresses safety in every aspect of operation. Conservation of resources for their intended purpose is a necessary activity when dealing with finite budgets. The potential for death and damage caused by robots, with behavior which can become too complex for the average person to predict, makes robots a powerful device that must be approached with extreme caution.

Summary

While an automated flightline initially appeared flighty, a closer examination shows that it may be very down to earth. The flightline has been shown to be a partially closed domain. The refueling task appears on the surface to be complex but well defined. While the goal to automate the task is ambitious, the chance to reduce the number of persons required to do the task offers a fall back position. The limitations are a combination of traditional engineering and AI constraints and as with all things connected with flying, safety must be the first consideration.

But these limitations are not the only obstacles in implementing a useful flightline robot. An easily expanded, flexible knowledge base will turn an interesting curiosity into a powerful useful tool. The flexibility the flightline demands requires a dynamic knowledge base. Automated knowledge acquisition is desperately needed and the next chapter presents the theory on learning necessary to discuss the knowledge acquisition task.

Theory

Introduction

From birth we spend our lives learning about the domain in which we live. We start with little or no knowledge. Through continuous exposure to input from our eyes we learn that the input is not random and we learn to pay attention to it. Continuous exposure to the set of features that make up our mother's face convinces us that she is important and we pay attention to her. Exposure to our hands and feet teaches us that they are under our control. Through both guided and unguided learning we grow from random motion to that which we are.

To help the reader understand learning this chapter will first trace through the general concept of learning. Next, the chapter will discuss the specific forms of knowledge acquisition that have been identified. Then, natural language understanding will be introduced as important element in learning. Finally, generalizing will be further elaborated on, and planning will be introduced as an important precursor to some forms of generalizing.

Learning Defined

Learning is defined as the process by which either a skill is refined or knowledge is acquired (6:6). Skill refinement predominately refers to a human becoming better able to perform a task through repeated execution of the task. An example of this is the penmanship lessons you performed as a child. Adaptive control theory is a machine version of skill refinement (6:6) and has been rigorously formalized. Knowledge acquisition, on the other hand, has not been formalized precisely.

Knowledge acquisition is the process by which a system obtains information that will allow it to perform a new task or to perform an old task in a new way. Improvement in task performance is sometimes stated as inherent in the learning process (6:6). This can be a matter of some confusion. Figure 3.1 from the Handbook of Artificial Intelligence shows a model of learning that includes a "learning element" and a "performance element" (9:327). By defining a subtask using a similar term for the task the chance for confusion is introduced.

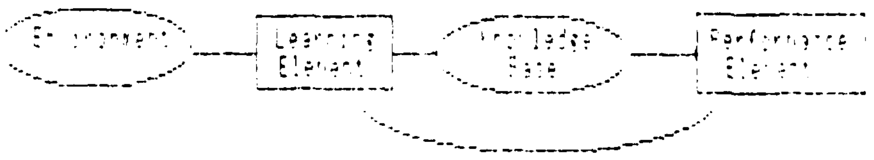


Figure 3.1 A simple model of learning systems (9:327).

The performance element of learning is critical in a completely autonomous system. To make this research problem tractable, the performance element will be ignored on the premise that the device under consideration is a semi-autonomous robot that will be supervised. It is believed that initially the supervisor will be needed in the feedback loop and that the performance element can be added to the system after the learning element is better understood. In this paper, unless clarity dictates otherwise, the learning element will be referred to as knowledge acquisition.

It is not yet understood if knowledge acquisition is fundamentally different from skill refinement or if they are superficially different manifestations of a similar biological process. The apparent tendency is for human biological systems to reapply useful processes to similar domains. For example, both sight and hearing use frequency and amplitude to measure perceptually different phenomena. This might lead one to believe both forms of learning involve similar events at some level. If this is true, the skill refinement related preceptron research of the recent past (9:378-379) and similar research going on today (15) may prove valuable in the knowledge acquisition field. As interesting as this possibility is, it is speculation at this point, and skill refinement will not be addressed further in this thesis.

Knowledge Acquisition

Knowledge acquisition can be divided into four types: rote learning, learning by being told, learning from examples, and learning by analogy (9:328). As each type of knowledge acquisition is presented, it is defined, and classical Artificial Intelligence (AI) program examples are presented. An example in the flightline robot domain is also presented. It should be understood that some events that are "obviously" knowledge acquisition will have characteristics that place them in more than one category.

Rote Learning.

Rote learning is best defined as direct transference of knowledge without transformation. It can be divided into learning by being programmed and learning by memorization, the difference being primarily whether the student or the teacher is expending the greater effort (6:8). In either case the distinguishing feature is the usable form in which the learning is presented. This distinction will become more clear as the other types of information are discussed.

Simplistic as it is, rote learning has some pitfalls. First is the amount of memory it consumes. Since transformation of information is not considered, numerous similar tasks do not share memory to store subtasks. Second is the "frame-problem." "Rote learning must be able to detect when the world has changed in such a way as to make the stored information invalid". Finally a truly intelligent system should be able to recognize that a solution is recomputable at less cost than saving the current solution (9:337).

The classic example of rote learning is L. A. Samuel's Checker Player. This program plays a game of checkers by looking ahead three moves and then using a static evaluation function to select the best move. The learning comes in when the static evaluation function can be replaced by a more accurate evaluation for a given situation. This more accurate evaluation is one that had been saved during a previous play of the same sequence. After saving about 53,000 positions the program was labeled by Samuel as "rather better-than-average novice, but definitely not ... an expert" (9:342).

Rote learning is also currently used in robots. Industrial robots are programmed using the "teach repeat" method of physically positioning the end effector of the robot and recording a number of intermediate "via points." By playing back the program, and using feedback from the same sensors used to record the program, servo motors can drive the end effector through the same motion that was recorded (15:12-59). In control theory parlance, this is an example of closed-loop control.

The Heathkit Hero-1 robot, an educational robot that served as the starting point for the AFIT MARRS robot [8,33], also makes use of rote learning. This system uses an open-loop recording mode where the input is from a remote control unit which the programmer uses to direct the robot through the desired path. After this, the program can be re-executed to command the same inputs to the robot. Note that, due to mechanical slippage and other errors, the same sequence of inputs may not generate precisely the same motion. A discussion of open-loop versus closed-loop control can be found in any good book on control system theory [11].

In the flightline robot domain, rote learning may be useful for simple tasks. In most cases the lack of feedback, as in the Hero-1 case, would not be sufficient, and even simple tasks would need to be broken up into subtasks and task completion indicators devised. Then the system could perform a subtask and verify that the subtask had been completed before attempting the next subtask. Rote learning might prove useful in defining primitives not envisioned by the original designer. The robot could be stepped through a small sequence of movements and this set of movements could be defined as a new primative.

Learning by Being Told. Learning by being told (advice taking) refers to information coming from an instructor that is too abstract or general to be directly used (9:328).

Instead it must be translated to a form that can be applied to the problem. Advice taking can be divided into five steps:

1) Request	- initiate transfer.
2) Interpret	- convert to internal form.
3) Operationalize	- make usable.
4) Integrate	- add to existing knowledge.
5) Evaluate	- determine usefulness (9:345).

This list addresses both the learning element, steps 1 through 4, and the performance element, step 5.

While no single work appears to have encompassed all of these, the most significant work on advice taking is Mostow's PhD thesis which addresses operationalization as the difficult task. In his First Operational Operationalizer (FOO) program, Mostow used about 200 rules to map advice on playing the card game Hearts into methods. Of note was that FOO did not include a planner to determine which rules to apply but depended on Mostow to provide the "control knowledge" to determine which rules to apply to accomplish the transformation (31:373).

Another significant work in advice taking that encompasses the other four steps is TEIRESIAS. To assist in maintaining an expert system knowledge base, TEIRESIAS takes rules in an English like form and transforms them into an internal representation. It also compares the information it was given to other rules to see if the rule appears to contain all the types of information that the other rules contain. If TEIRESIAS finds a category of information omitted it asks if the difference was intentional. (41:199-200)

An example of learning by being told in the flightline robot domain might be "Refueling a C-141 is just like an F-16 except that C-141s take a lot more fuel and do not have hydrozine on board." In this case the robot might have to operationalize the hidden information that a large truck is preferred since the process might take many hours using small fuel trucks. Since hydrozine is not on board it might conserve effort by ignoring some small number of safety precautions that only relate to the presence of hydrozine.

Learning from Examples. Learning from examples, also called generalization or induction, involves a system, given examples and possibly counter examples, extracting some commonalty to arrive at some higher level concept. The examples may come from a teacher, from the learner (experimentation), or from the environment (observation). The learner can be either supervised or unsupervised (6:9-11).

One example of this was BASEBALL, a program which took about 2000 "snapshots" (descriptions of various stages of the game) and derived the rules of baseball (9:364). Other examples are BACON, which takes numeric relations and "discovers" physical laws (26:307-330), and ARCH, which takes positive and negative instances of an arch formed of blocks and derives the concept of an arch (32:385-408).

As an example in the flightline robot domain, the robot would observe the usual steps in preparing an aircraft for flight and to develop the concept of flight-worthy. Another example that will form the basis of work later in this thesis is for a flightline robot to take examples of specific instances of refueling aircraft and generating a general concept of refueling aircraft.

Learning by Analogy. Learning by analogy involves "acquiring new facts or skills by transforming and augmenting existing knowledge that bears strong similarity to the desired new skill into a form effectively useful in the new situation" (6:8). Little work has been done in the area of learning by analogy. A large amount of world knowledge is needed to estimate that a new task is similar enough to a known task to allow the use of existing techniques in the new domain. No one has suggested a formal mechanism for how humans perform analogies. We need to understand much more about the process before we can hope to program a machine to draw analogies.

No examples of analogy drawing programs are available. The only sample that can be presented is an example of man doing the task. A person may attempt to drive a truck based on his experience with a car. With luck, the driver may be successful but if the truck is large and the driver does not consider a truck as requiring greater stopping distance than a car, the driver will likely get in trouble. This is to say that analogies may or may not be successful.

An example in the flightline robot domain would be for a robot to fill a hydraulic system reservoir based on knowledge about fueling an aircraft.

To summarize, Figure 3.2 depicts learning and its subparts. With the general concept of learning understood, two additional topics must be introduced to allow the design to be understood. A brief discussion of natural language understanding is presented and then a discussion of the relationship between generalization and planning follows.

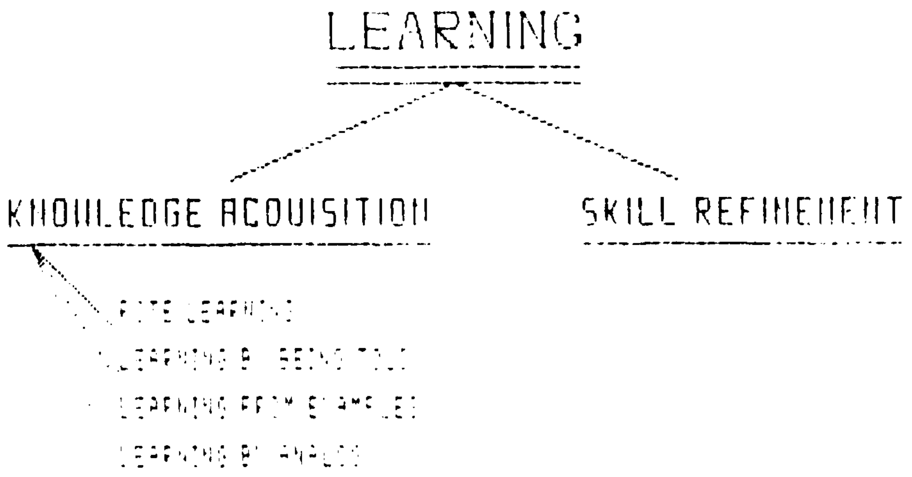


Figure 3.2 A taxonomy of learning.

Natural Language Understanding

Earlier, the model in Figure 3.1 depicted the need to take information from the environment and present it to the learning element. Often the information in the environment originates from another individual and exists the form of speech or writing. In both instances man has developed a representation of thoughts in the form of symbols that he has found easy to communicate and understand. It is apparent from studying the brain that the internal representation of these thoughts is very different. In the case of a computer, the form of the internal data and the manipulations that can be done are well known. The goal of natural language understanding is to take the information in the environment and perform a translation, not unlike that which the brain does, to put it in a form that can easily be operated on by the computer.

In Scripts Plans Goals and Understanding (36) Schank reviews and expands upon his earlier work on natural language understanding through the use of Conceptual Dependencies (CDs). Stated simply, the Conceptual Dependency form of representation uses the verb of a sentence to determine what additional information the sentence should contain. It attempts to map all verbs into a subset of verbs with each element of the subset representing a unique concept. For instance, walk and run are conceptually the same to Schank in that the result is a physical transference of some person. Having found a "PTRANS" verb such as walk, one can expect to find an actor who does the walking and possibly a origin point and destination point.

Consider the two sentences "Jack walked to the store." and "Jill ran from the house." These are conceptually the same in that in both cases a person was physically transferred from one place to another:

```
(PTRANS (Actor Jack) (Source ?) (Destination Store))
(PTRANS (Actor Jill) (Source House) (Destination ?))
```

This simple example of CD representation also shows some of the weaknesses of the method. While the actions are similar, standard CD representation ignores questions of rate and time in an effort to simplify things. It also tends to ignore intent. Certain groups of words establish a mind set that often helps resolve the meaning of a sentence or group of sentences. Here it is important to begin to assume that Jack wanted something while Jill may have feared for her life. This may prove not to be the case but understanding intent provides additional information. It is not desirable to expect a computer to add information which is not in the original sentence, but it is also not desirable to delete information that may resolve meaning later.

Schank addresses this by describing the building of "semantic nets." Rather than pursue this, since the flightline should only require a restricted subset of English for communication, a simpler method of including the reduced concepts as well as all of the initial details should prove sufficient. After this interpretation process is complete the computer can perform some transformation on the text. Generalization is one such transformation.

Generalizing and Planning

Programming a machine to generalize would provide at least two advantages. The less important advantage would be the conservation of memory required to store world knowledge. Consider a system that could reorganize its memory and that could recognize that two or three of the plans it has are similar and differ only in a few places. These plans could be merged into a single plan that has variables substituted for some of the constants in the original plans. This is analogous to a programmer writing a computer program that uses subroutines to perform similar repeated computations.

The more important advantage would be in the time saving when the robot is to be taught a new task. Given that the robot knows how to refuel a number of aircraft, if it has also performed the above reorganization, it has a crude script to follow while being taught how to handle a new aircraft. It could follow the instructor along and tell the instructor that, for example, it already knows how to remove the fuel cap but needs to be shown where the fuel cap is located. At this point it also knows how to insert the fuel nozzle, and so forth. Rather than starting from scratch, only the new and unique aspects of the task need to be taught. This offers a great speed up in programming time for both the robot and the teacher.

A difficulty in generalizing high level plans is that the similarities can be disguised in the high level statement of the plan. If one plan states "Ground the refueling equipment." and the other states "Connect the static bond cable to the aircraft and connect the fuel source ground wire to the grounding point." the similarity might be lost without the world knowledge of either the method to accomplish the first or the purpose for accomplishing the second. One way to include this world knowledge is to take both statements of the task, determine the steps involved in each, and then compare the steps involved. To accomplish this requires the use of a planner. One of the better known planners to date is the Stanford Research Institute Problem Solver (STRIPS). Moravec recently called STRIPS "... a powerful, effective, still unmatched system..." (30:127). It serves as an excellent example to study.

STRIPS is a program that uses three structures to operate. First, it requires a world model to describe the state of the world in which planning is to be done. Second, it requires a list of operators that describes what capabilities it has to alter the world state. Finally, it requires a goal state that is desired in the world model.

The world model is represented by predicate calculus well formed formulas (wffs). These wffs describe discrete micro states. For example, NEXTTO(ROBOT,BOX1) indicates that among other things that are true in the world, the robot is next to box1. The first portion of the sample wff, NEXTTO, is called the predicate. The portion within the parenthesis are constants or variables whose meaning is dependent on the predicate. A more through treatment of predicate calculus can be found in a text on the subject [7] but this explanation should be sufficient to understand material presented here.

Figure 3.3 is a reproduction of the original STRIPS world that is to be modeled. Table 3.1 reproduces the initial world model stated in predicate calculus wffs. The STRIPS operators describe how the planner can alter the world state to achieve some goal. This is done by providing the name of the operator, the instantiated variables, preconditions to applying the operator, and a delete list and add list that indicate how the world model is changed. Table 3.2 lists the available STRIPS operators.

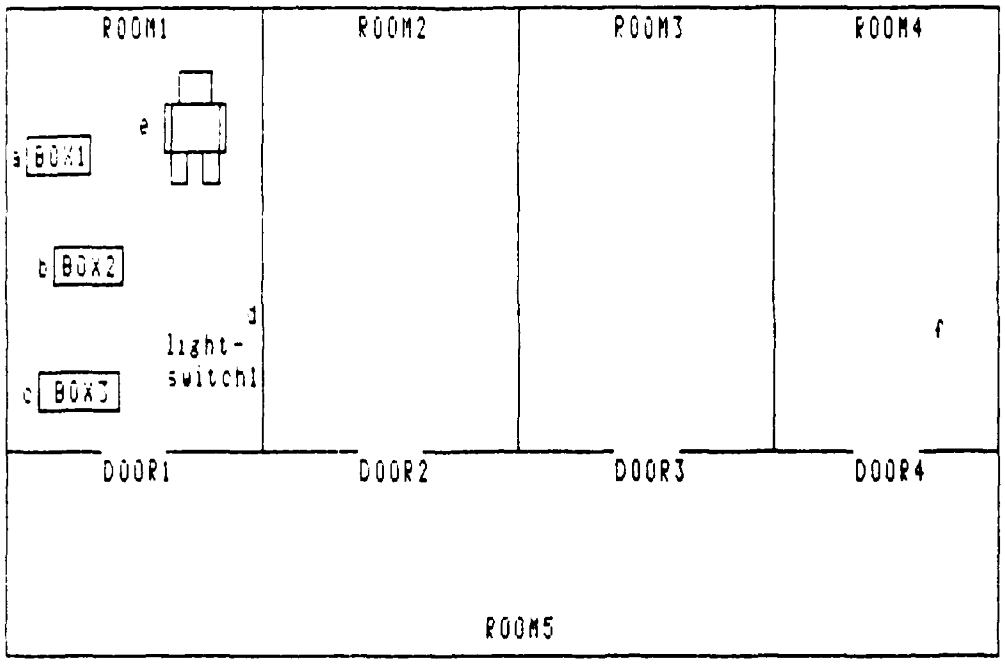


Figure 3.3 Room plan for the robot tasks (18:203).

TABLE 3.1

STRIPS Initial World Model (18:204)

CONNECTS(DOOR1,ROOM1,ROOM5)	CONNECTS(DOOR1,ROOM5,ROOM1)
CONNECTS(DOOR2,ROOM2,ROOM5)	CONNECTS(DOOR2,ROOM5,ROOM2)
CONNECTS(DOOR3,ROOM3,ROOM5)	CONNECTS(DOOR3,ROOM5,ROOM3)
CONNECTS(DOOR4,ROOM4,ROOM5)	CONNECTS(DOOR4,ROOM5,ROOM4)
LOCINROOM(f,ROOM4)	ONFLOOR
AT(BOX1,a)	INROOM(BOX1,ROOM1)
AT(BOX2,b)	INROOM(BOX2,ROOM1)
AT(BOX3,c)	INROOM(BOX3,ROOM1)
AT(LIGHTSWITCH1,d)	INROOM(LIGHTSWITCH1,ROOM1)
ATROBOT(e)	INROOM(ROBOT,ROOM1)
TYPE(BOX1,BOX)	PUSHABLE(BOX1)
TYPE(BOX2,BOX)	PUSHABLE(BOX2)
TYPE(BOX3,BOX)	PUSHABLE(BOX3)
TYPE(LIGHTSWITCH1,LIGHTSWITCH)	STATUS(LIGHTSWITCH1,OFF)

TABLE 3.2

STRIPS Operators (18:204-205)

```
[1] gotol(m): Robot goes to coordinate location m.
 Preconditions: ONFLOOR & INROOM(ROBOT,x) & LOCINROOM(m,x)
 Delete list: ATROBOT($),NEXTTO(ROBOT,$)
    Add list: ATROBOT(m)

[2] goto2(m): Robot goes next to item m.
 Preconditions: ONFLOOR &
                {[INROOM(ROBOT,x) & INROOM(m,x)] |
                 [INROOM(ROBOT,x) & CONNECTS(m,x,y)]}
 Delete list: ATROBOT($),NEXTTO(ROBOT,$)
    Add list: NEXTTO(ROBOT,m)

[3] pushto(m,n): Robot pushes object m next to item n.
 Preconditions: PUSHABLE(m) & ONFLOOR & INROOM(ROBOT,m) &
                {[INROOM(ROBOT,x) & INROOM(m,x)] |
                 [INROOM(ROBOT,x) & CONNECTS(m,x,y)]}
 Delete list: ATROBOT($),NEXTTO(ROBOT,m),
              AT(m,$),NEXTOOmmmm),NEXTTO(m,$)
    Add list: NEXTTO(ROBOT,m),NEXTTO(ROBOT,n),
              NEXTTO(m,n),NEXTTO(n,m)

[4] turnonlight(m): Robot turns on lightswitch m.
 Preconditions: TYPE(x,BOX) & ON(ROBOT,x) &
                NEXTTO(x,m) & TYPE(m,LIGHTSWITCH)
 Delete list: STATUS(m,OFF)
    Add list: STATUS(m,ON)

[5] climbonbox(m): Robot climbs up on box m.
 Preconditions: ONFLOOR & TYPE(m,BOX) & NEXTTO(ROBOT,m)
 Delete list: ATROBOT($),ONFLOOR
    Add list: ON(ROBOT,m)

[6] climboffbox(m): Robot climbs off box m.
 Preconditions: TYPE(m,BOX) & ON(ROBOT,m)
 Delete list: ON(ROBOT,m)
    Add list: ONFLOOR

[7] gothrudoor(k,l,m)
 Preconditions: NEXTTO(ROBOT,k) & CONNECTS(k,l,m) &
                INROOM(ROBOT,l) & ONFLOOR
 Delete list: ATROBOT($),NEXTTO(ROBOT,$),INROOM(ROBOT,$)
    Add list: INROOM(ROBOT,m),NEXTTO(ROBOT,k)
```

Three points require further explanation. First, any precondition that does not exist in the model can be made to exist if an operator can be found to do the task. The simple solution would be to try all operators. As the number of operators gets large this becomes a large search problem. This is reduced by only trying those operators that contain the desired predicate in their add lists. Second, any variable within the operator can take on any value consistent with the world model, provided it takes on the same value for the whole operator. Two variables may have the same value if this is consistent with the world model. Third, the dollar sign "$" is used to represent a "wild card" match in the delete list. This variable can be matched with any constant in a wff in the model.

To understand how this operates, consider a simple goal "ON(ROBOT,BOX1)." In this case STRIPS checks the world model and determines that the robot is not on BOX1. It checks the available operators and finds CLIMBONBOX is the only operator that might prove useful in trying to satisfy the goal. It then checks to see if the preconditions for this operator are true. The first two, ONFLOOR and TYPE(BOX1,BOX) are true but the third, NEXTTO(ROBOT,BOX1) is not. Again STRIPS looks for an operator to make this true and again only the operator GOTO2 is possible. Since all the preconditions of GOTO2 are true, STRIPS applies the operator by deleting items from the world model that GOTO2 calls to delete (ATROBOT(e) matches ATROBOT($)) and adds items that it calls to add (NEXTTO(ROBOT,BOX1)). Now all of the preconditions of CLIMBONBOX are satisfied and again the world model can be updated. The resulting plan is GOTO(BOX1), CLIMBONBOX and at the completion of the execution of this plan the world model would contain the wff ON(ROBOT,BOX1).

To see how this can be used by a generalizer, consider the two operators TURNONLIGHT and CLIMBONBOX. At first these items appear to have little in common. TURNONLIGHT has some preconditions common with CLIMBONBOX but not much can be said about the commonalty of the two. After planning TURNONLIGHT contains the steps GOTO(BOX1) PUSHTO(BOX1,LIGHTSWITCH) CLIMBONBOX(BOX1) TURNONLIGHT(LIGHTSWITCH1). After planning CLIMBONBOX contains the steps GOTO(BOX1) CLIMBONBOX(BOX1). Comparing these two plans shows GOTO(BOX1) CLIMBONBOX(BOX1) common to both. Similarities between the two become apparent after planning. It is with the idea that this information might be useful that a planner and a generalizer are combined to accomplish learning.

Summary

While learning is not completely understood, a taxonomy of learning has been developed to allow discussion of what are perceived to be different aspects of learning. The division of learning into a learning element and a performance element allows each of these to be addressed separately.

A taxonomy based on types of learning separates learning into skill refinement and knowledge acquisition. A further division of knowledge acquisition into rote learning, learning by being told, learning from examples, and learning by analogy facilitates discussion. However the actual differences in mechanism, if there is one, is unknown. That one does not exists is supported by some instances of learning which do not fall squarely into one category.

Learning cannot be discussed in a vacuum and so this chapter has also included introduction to natural language understanding and planning. A design to tie these pieces together is presented in the next chapter.

Design

Knowledge acquisition encompasses almost every other AI topic. The design of a project at the thesis level must restrict itself to a subset of the topic. This thesis has three main thrusts. First is the transformation of the two available definitions of the task to an internal form. Second is planning to discover unspecified portions of the task. Third is to combine the two instances to discover their commonalty.

Task

The specific task to be accomplished is to take the refueling Job Guides for the F-15 and the F-16 as the input to the system and produce as output a generalized script that might be useful in understanding actions in any aircraft refueling process. The normal refueling Job Guides for these two aircraft are reproduced side by side in Appendix A. Major headings and conceptually equivalent steps are aligned to make comparison easy. This provides the reader an opportunity to get a feel for their degree of commonalty.

The skeptic reader should immediately ask two questions. The first question he should ask is why only two aircraft? The complexity of the problem demands simplification to allow progress to be made in the available time. Also, time required to handle additional aircraft would detract from other aspects of the thesis. The second question the reader should ask is why these two aircraft? Originally an attempt was made to compare the C-141 and the F-16 refueling Job Guides. Similarities exist between these two aircraft but the C-141 Job Guide contains page after page of material that is very aircraft specific. The sheer volume of material clouded the issue and the effort required to present all of this data only to have it deleted when extracting similarities did not seem constructive. The reader unfamiliar with the F-15 and F-16 should realize that these two aircraft, while somewhat similar, are produced by different manufacturers and it is the manufacturers that write the Job Guides.

The task can be subdivided into three subtasks. The first subtask is the transformation of the refueling Job Guide steps into Conceptual Dependency (CD) form, the application of domain and world knowledge, and the transformation of the result into well formed formulas (wffs). This is done to provide the second subtask, planning, with a form that has proven successful in many planners. The second subtask it to take the wffs and, by planning, produce an operational script of primative operators. The third subtask is to compare two operational scripts to extract the essence of the general refueling task.

Transformation

The form of the refueling Job Guide is primarily standard written text. As noted before, the form required by the planning stage is predicate calculus wffs. Three phases are used in this step to transform the text into wffs. First, Schank's work is used to transform the text into a CD representation. Second, using expert system techniques like Mostow's, the CDs can be operationalized to generate an extended CD representation. This form was designed to include details normally left out of the CD form. Finally, the extended CDs can be examined by another expert system to determine which wff to apply. An example of this transformation is presented in the next chapter.

Since terminology is different from aircraft to aircraft, the system will require a large knowledge base. Many terms are stated in less detailed terms after they are introduced. For example, while "fuel nozzle" is referenced early in the F-15 Job Guide, later references are simply to the "nozzle." Further, some terms are specified only within the context of the rest of the sentence or the rest of the process description. An example of this is the aircraft refueling adapter cap is simply referred to as the "cap" in the sentence "Remove cap from aircraft refueling adapter." A large body of world knowledge is required to successfully make these inferences.

Planning

A planner is needed to accept the list of wffs generated by the transformation process and to produce a complete script of primative operators needed to perform the task. The STRIPS planner that was described in the previous chapter is used as a model. STRIPS lacks at least five features that would prove useful as the domain becomes more complex.

Some tasks require post-application conditions. These clean-up actions are not directly provided for in STRIPS. One example might be closing a door when you leave the room. The modified "gothrudoor" operator is shown:

```
gothrudoor (k,l,m): Robot goes through door k from room l
                    to room m and closes door k behind him.
  Preconditions:
    NEXTTO(ROBOT,k) & CONNECTS(k,l,m) &
    INROOM(ROBOT,l) & ONFLOOR
  Postconditions:
    STATUS(k,CLOSED)
  Delete list: ATROBOT($),NEXTTO(ROBOT,$),INROOM(ROBOT,$)
     Add list: INROOM(ROBOT,m),NEXTTO(ROBOT,k).
```

This additional entry in the description of the operator should not be mandatory for all tasks.

Application of operators, in general, have both effects and side effects. As presented, STRIPS lumps these together in the add or delete lists. Consider the "pushto" operator. The add list of that operator was originally defined as:

```
Addlist: NEXTTO(m,n)
         NEXTTO(n,m)
         NEXTTO(ROBOT,m).
```

If the planner has the goal of the robot being next to box2, STRIPS might select pushto(box1,box2) and at the completion of this operation the robot is nextto box2. One solution is to divide the add list into effects and side effects. This leaves the full power of STRIPS but allows the separation and a reduced search space when desired. Thus the add list of the "pushto" operation becomes:

```
Addlist: NEXTTO(m,n)
         NEXTTO(n,m)
         +                        <-- delimiter
         NEXTTO(ROBOT,m).
```

This representation explicitly states that the effect of applying the operator is to get m next to n and n next to m while the fact that the robot is next to m at the completion of the operator application is simply a side effect.

STRIPS Operators generally have preconditions that, if not satisfied in the current world, become candidates for being made true through the application of other operators. A useful extension to this is to allow the designer to establish preconditions that, if not satisfied in the current world, remain unsatisfied. This has two applications. First, it speeds up execution when applied to preconditions that will never be made true. There is no operator that can be used to make "TYPE(BOX1,DOOR)" and it would be smarter not to look for one. The second use would be when an operator would need to retrieve some information about a state in the world that it can change but does not want to change. This application is beyond the level of sophistication of the original STRIPS operators. Consider an operator that does one thing when "HANDEMPTY" is true and another when "GRASPED(m)" is true. If something is grasped, the planner should not discover that "HANDEMPTY" is not true but can be made true through the "release" operator.

While providing for the application of operators to make things true that are not true, STRIPS does not provide a method of making things not true when that is desired. This ability would allow for two conditions. The size of the world model would be reduced. Rather than requiring "HANDEMPTY" as an entry in the world model, "~GRASPED(x)" (where ~ stands for not) could be tested for. This would also allow preconditions to be stated more like people think. The only requirement to implement this is that in addition to being able to search the add lists of operators when a condition is desired, the program needs to be able to search delete lists of operators when an existing condition is not desired.

The final addition to STRIPS is the addition of a critic. This enhancement provides for different semantics in interpreting the goal:

GOAL-1 and GOAL-2.

The first interpretation is: "Do GOAL-1 and when you are done, do GOAL-2." The second interpretation is: "Develop a plan such that when you are done applying the plan GOAL-1 and GOAL-2 are true." Consider the two cases:

CASE 1: NEXTTO(BOX1,BOX2) & NEXTTO(BOX2,BOX3).

The first plan that the planner might suggest is:

```
GOTO(ROBOT,BOX1),
PUSHTO(BOX1,BOX2),
GOTO(ROBOT,BOX2),
PUSHTO(BOX2,BOX3).
```

If the intent of the user was to get all three boxes together, the planner has failed. By adding a critic to verify that all of the original sub-goals are true in the

//

final world model, the critic can find the failure, reject this plan, and let the planner try to come up with another plan.

On the other hand, consider the following:

```
CASE 2: NEXTTO(ROBOT,FUEL-SOURCE) &
        GRASP(FUEL-HOSE) &
        NEXTTO(ROBOT,AIRCRAFT-REFUEL-POINT).
```

Here we can guess that the user is providing a sequence of steps that are part of some refueling task and probably does not intend for the robot to move the fuel source closer to the aircraft refuel point. The application of the critic in this case is not desired.

A function to replace the critic could be embedded in the main planner and a separate syntax developed for both types of conjunctive goals reducing planning time by attempting to generate the desired type of plan from the start. This was not implemented.

While STRIPSwwss a powerful planner with an elegant simplicity of representation, these enhancements to STRIPS will provide an even more flexible planner that will handle even more complex tasks.

Generalizing

Given the output of the planner for two different refueling Job Guides, the purpose of the generalizer is to extract some commonalty in the two plans. A simple intersection of the two sets of primative operations would fail to include the information contained in the order in which the operations are performed. The commonalty that will be considered is the longest list of primative operations that is common to both lists while maintaining sequential ordering. Table 4.1 presents some examples of this concept.

TABLE 4.1

Sequential Commonalty

List 1	List 2	Commonalty
(A B C)	(A B D)	(A B)
(A B C D)	(A C D B)	(A C D)
(A B C D E)	(B C A D E)	(B C D E)
(A B A B C)	(A B C A B)	(A B C)
(A B C D E)	(C D A B E)	(((A B) (C D)) E)

In the last case there are two lists that have equal lengths and therefore both must be returned. This notation indicates both (A B E) and (C D E) are valid solutions.

Thus the premise is that a large number of similar steps can be found in two conceptually related plans, that it can be extracted, and that it represents the essence of the overall concept. At this point the reader is encouraged to examine Appendix A closely. This represents a real world example of the recommended method to refuel two different aircraft. The tasks have numerous common steps in before planning. After the planner fills in steps that one Job Guide author assumed given while the other stated explicitly the plans will exhibit even greater commonalty.

Another means of comparing two operations is to examine their effect on the world model. While an analysis of the world model before and after the application of separate refueling scripts will show that in both cases the quantity of fuel in the aircraft has changed, that is all it would show. If a robot is to learn how to refuel another aircraft with this as a starting point, it still has a lot to learn. If however, a robot has a plan that lists the core steps in refueling other aircraft, it has a larger framework to build on.

The algorithm for performing this will be presented in the next chapter.

Implementation

The art of transforming concepts into working systems is the central core of engineering. Engineering is implementation. The implementation of the design concepts of the last chapter is the topic of this chapter. This process involves drawing together several contemporary AI programs that have been described elsewhere in this paper.

The design in the last three sections of the previous chapter is implemented here in a corresponding section. The first section describes how to transform the refueling Job Guide text into predicate calculus well formed formulas (wffs). This internal representation is used since nearly all successful planners have used wffs to represent the world model. The second section describes how to take a script of wffs and, through planning, produce a script of primative operators. This fully operationalized plan then can be compared to other plans. The third section describes how to derive a generalized plan from multiple instances of similar plans.

Transformation

Transformation of a refueling Job Guide entry into a predicate calculus wff is not a trivial task. To demonstrate the feasibility of the task, a three step transformation can be applied. The number of steps was chosen based on examination of the input text and knowledge of working AI systems. Systems exist to perform the three transformations and thus the only requirement is to demonstrate that these can be tied together to work in the flightline domain. Combining steps, while feasible, would involve a tighter coupling (interaction) of the systems. Traditional software engineering principles discourage this. Decomposing the steps further was not examined.

The first step falls under the heading of natural language understanding. Drawing primarily on the work of Schank [38], the refueling Job Guides will be transformed into a Conceptual Dependency (CD) like form. The form used, while varying slightly from Schank's CD representation, maintains most of the character and flavor of his work. The difference will not be considered in this paper and the purist can hereafter think "modified CD" when the term "CD" is encountered. Consider the following step taken from the F-15 refueling Job Guide:

4. (B) Remove ground refueling receptacle cap.

This is the fourth enumerated step in the refueling task. It is to be performed by the person filling the role of "man B." This is not standard English syntax but it is consistent throughout the procedure and can be accommodated. The general responsibilities of each person are covered in the preface to the refueling task. This is included in Appendix A. The text portion of the refueling task is self explanatory.

In CD representation this becomes:

```
(MOVE (VERB   ((SPECIFIC remove)))
      (ACTOR  man-B)
      (OBJECT refueling-receptacle-cap)
      (SOURCE nil)
      (DESTIN nil)).
```

Notice that while remove has been simplified to the Conceptual Dependency concept "MOVE," the original verb is retained to provide details for those conditions that require specifics. Notice also that many details are missing. The place from which the fuel cap is to be removed is not given and neither is the destination. The method required to accomplish the task is also assumed to be known.

General world knowledge is required above and beyond that expressed in the CD form. Mostow's work [31:367-403] and work with micro PAM [37:181-196] demonstrated how transformations can be applied to provide this knowledge. Since Job Guides are written to be used by people with varying degrees of expertise, they are written explicitly. A radical transformation of the instructions should not be required in such a well defined environment. For this reason an "extended Conceptual Dependency" form is proposed and employed to represent the operationalized CD. This form retains all of the detail of the original CD to cover those cases where that level of detail is required. It also contains general knowledge in the form of simplifications, consistent terminology, domain dependent information, and context dependent information. After the application of the operationalizer, the extended CD becomes:

```
(MOVE (VERB   ((SPECIFIC remove)))
      (ACTOR  man-B)
      (OBJECT fuel-cap(F-15)
              ((SPECIFIC refueling-receptacle-cap)))
      (SOURCE SPR(F-15))
      (DESTIN stowage-point)
      (ORGAN  hand)).
```

The operationalizer looked at MOVE and determined that the meaning in this domain could be verb specific. It looked at remove and found again that the method was object specific. When it examined the object it found a non-standard representation and substituted the standard representation. It examined the standard representation and determined that the combination of remove and fuel cap was sufficient to define a primative operation involving the hand. It looked at the remainder of the CD and found default value for the source is the single point refueling adapter (SPR) and the default value for the destination is a storage bin on the robot.

The third transformation on the Job Guide entries was to change the extended CDs into predicate calculus wffs. This step can also be done by an expert system. It is not combined with the previous transformation because of the different relationship between the input and the output and to keep the general world knowledge separate from the knowledge the robot has about states that can exist in the world model. The mapping of the previous example to wffs produces:

```
extract(fuel-cap(F-15) SPR(F-15))
move(fuel-cap(F-15) stowage-point).
```

With these transformations complete, the information is now in a form that can be operated upon by a planner.

Planning

By combining the individual wffs into a list, a script is obtained that represents refueling of that particular aircraft. But such a list is not complete. The planner designed in the last chapter is implemented here in the form of PostSTRIPS. PostSTRIPS gets its name from its similarity to the STRIPS program it is modeled after. To distinguish it from STRIPS it is called PostSTRIPS because of its provision for postconditions in its operators. PostSTRIPS is implemented in LISP. Without the source code for STRIPS, it was not possible to duplicate the internal operation of STRIPS. The next few paragraphs provide a detailed description of the operation of PostSTRIPS.

PostSTRIPS uses a double ended queue (deque) of nodes as its primary control structure. In a deque, nodes can be added (pushed) or removed (popped) from the top or bottom.

An initial node is created that consists of five lists. The world model consists of a list of wffs. The goal stack consists initially of the single wff that is to be planned for. The binding list, initially nil, will hold an association list. This list consists of variable and value pairs as each variable is instantiated while considering a particular solution path. The operator list, initially nil, will hold each operator, in sequence, that is the candidate for establishing a particular goal. Finally, the plan list, initially nil, receives each operator as all of its preconditions are met. The program operation can best be stated in a structured English format.

To plan for a desired goal:

1. Form a deque with one node consisting of the current world state, a goal stack, a binding list, an operator list, and a plan list.

2. Until the deque is empty or the goal stack is nil:

2a. Pop the top node from the deque.

2a1. If a conjunctive compound goal is encountered, push each subgoal on the goal stack and push the new node on the top of the deque.

2a2. If a disjunctive compound goal is encountered, push a node on the top of the deque for each subgoal after adding that subgoal to the remaining goal stack.

2a3. If the top goal flags success in applying an operator's preconditions; apply operator to the world model, pop flag from goal stack, pop operator from operator list, instantiate operator (with binding list), push operator on plan list and push node back on the top of the deque.

2a4. Otherwise, if the top goal flags that all operator conditions have been met, and a test shows this world state did not already exist on this solution path; pop the flag, and push the node on the top of the deque.

2a5. Otherwise, if the top goal flags having tried all possible bindings in the current world, push a node on the top of the deque for each operator in the operator list that might make the goal true, unless an examination of the goal stack indicates that this is a reoccurring goal. Before pushing the node on the deque, add the operators preconditions, a success flag, postconditions and a flag to indicate all operator conditions have been met to the goal stack, and the operator to the operator list.

2a6. Otherwise, if the top goal in the node is absolutely true (no bindings need be made); delete the top goal from the goal stack and push the node on the top of the deque.

2a7. Otherwise, if the top goal in the node is conditionally true; push one node on the deque for each set of bindings that make it true. Before pushing the node on the deque, delete the top goal from the goal stack and add the bindings to the binding list. Also push a node on the bottom of the deque reflecting the original goal but flagged to indicate that all possible bindings have failed to provide a solution, unless this goal is flagged as one that must exist in the current world.

2b. If any nodes remain, remove the top node and critique the plan.

2b1. If successful return the plan.

2b2. Otherwise, continue with remaining nodes.

3. If found, return the plan; otherwise, announce failure.

Note: A reoccurring goal is evidence of a nonconverging condition that cannot produce a solution.

Selection of applicable operators is eased by an association list of wff predicates and operators that contain that predicate in the effects portion of the operator add list. PostSTRIPS consults the add list to see if it is possible to unify the two wffs. If so, the operator is tried. This is PostSTRIPS implementation of means-end analysis.

The critic is implemented by using the same deductive retriever that the rest of the routine uses. It simply takes the original goal and tries to retrieve it from the new world model. Failure indicates that some subgoal was deleted while establishing a subsequent subgoal.

In addition to implementing PostSTRIPS, a world model to represent the flightline domain is needed. Figure 5.1 depicts the world that is to be represented.

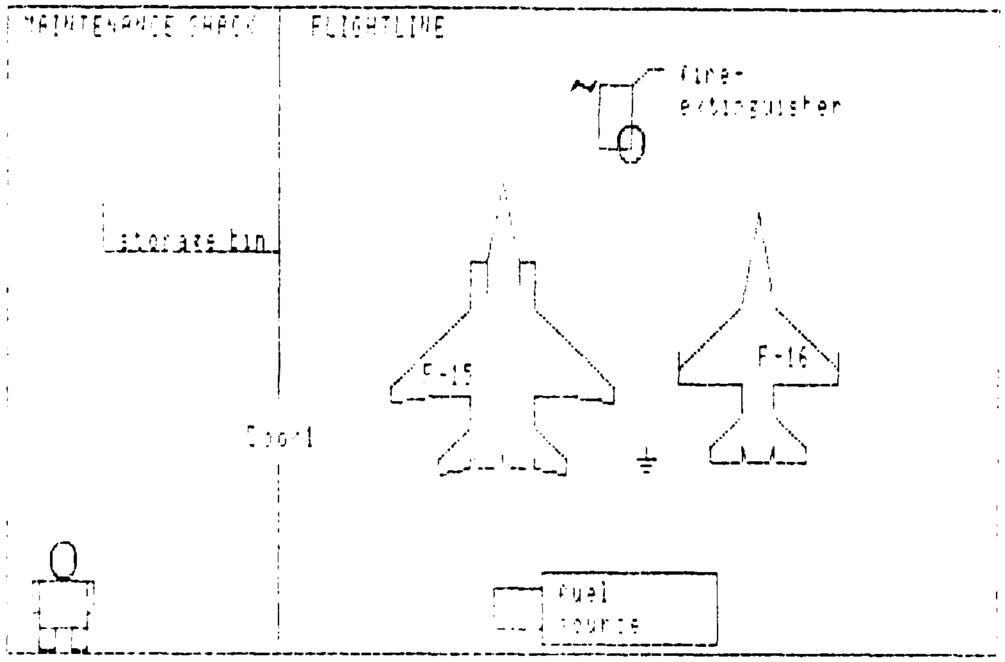


Figure 5.1 Initial PostSTRIPS refueling domain

Finally, a set of operators that are applicable to the flightline domain are needed. By examining the refueling task, and by estimating the mechanical abilities of a flightline robot, a set of operators relating the two can be coded and tested. Some of these will be very similar to the original STRIPS operators, however, the original STRIPS operators did not reflect a very complex robot. These operators will be discussed more in the next chapter where they are presented along with the representation of the world model and the results of testing.

Generalizing

The last chapter described a sequential commonalty operator to find the essence of two lists. This longest list that is common to both with sequential ordering maintained is conceptually not difficult to obtain. The following describes the procedure.

To find the longest common sequence in two lists:

```
1. Establish the solution as nil.
2. Until one list is nil, compare the first elements of each
list:
   2a. If they are equal, add the element to the solution
and delete it from the lists.
   2b. Otherwise, if only one occurs in the other list,
remove elements up to the occurrence in the list.
   2c. Otherwise, delete the sublist containing them,
compute both possible solutions and:
        2c1. If one is longer, add it to the solution.
        2c2. Otherwise, add both to the solution.
3. Return the solution.
```

The difficulty with this solution is that with longer lists, the time required to calculate the solution exceeds the time available to do the research (see next chapter). An alternate solution can be obtained by using the original pre-planning scripts to suggest way points at which the problem can be divided. The following describes the procedure.

To estimate the longest common sequence in two lists:

```
1. Establish the solution as nil.
2. Propose a list of elements to segment the lists into smaller lists. For each pair of smaller lists:
   2a. Until one list is nil, compare the first elements of each list:
       2a1. If they are equal, add the element to the solution and delete it from the lists.
       2a2. If only one occurs in the other list, remove elements up to the occurrence in the list.
       2a3. Delete the sublist containing them, compute both possible solutions, and:
            2a3a. If one is longer, add it to the solution.
            2a3b. Otherwise, add both to the solution.
3. Splice smaller solutions into larger single solution.
4. Return the solution.
```

This solution appears to reduce the computational explosion. The results of the execution can be found in the next chapter.

Results

No design is complete until its implementation demonstrates the designed purpose. Paralleling the format of Chapter 4 and Chapter 5, this chapter will present the result of the implemented design. The first section presents the results of transforming the F-15 and F-16 Job Guides into well formed formulas (wffs). The second section provides both a comparison of PostSTRIPS and STRIPS in the original STRIPS domain and the results of PostSTRIPS in the flightline domain. The third section explains difficulties encountered with the first generalizer and then explains how the second generalizer overcame these difficulties.

Transformation

Schank covered the subject of natural language understanding in considerable detail [38]. Since the major thrust of this thesis is not natural language understanding and since trying to recreate Schank's work would detract from time available for other aspects of this thesis, a natural language interpreter was not implemented. Instead, the author drew upon prior experience with a natural language understander [37:181-196] and converted the F-15 and F-16 refueling Job Guides into Conceptual Dependencies (CDs) by hand.

A small expert system was built to demonstrate how the transformation of CDs to extended CDs might be accomplished. Mostow's work in this area describes the kind of system needed [31:367-403] and a complete system to implement all of the rules would have been too time consuming. A considerable amount of world knowledge was applied during the conversion process.

The third transformation from extended CDs to wffs can be done by a process similar to that used to convert CDs to extended CDs. The results of transforming the F-15 Job Guide steps into wffs, including the intermediate CD and extended CD representation, is presented in Appendix B. Similar results for the F-16 can be found in Appendix C.

Planning

Operation of the planner was investigated in two areas. First, to verify the operation of the program, the less complex domain of the original STRIPS world and original STRIPS operators formed the basis for testing. Second, to show the operation of PostSTRIPS on the flightline refueling task, an extension to the original STRIPS world and operators was devised and tested.

To compare the operation of STRIPS and PostSTRIPS, the original STRIPS world and operators were transcribed with one change to the "turnonlight" operator that will be discussed later. The reader is referred back to Chapter 3 for an explanation of the world and operators. PostSTRIPS was given the same tasks to plan for that were described in the original STRIPS report [18:205]. Table 6.1 compares the plans produced by STRIPS to those produced by PostSTRIPS.

TABLE 6.1

Comparison of STRIPS and PostSTRIPS Plans

```
Tasks

1. Turn on the lightswitch.
   Goal wff: STATUS (LIGHTSWITCH1, ON)
   STRIPS solution:
             {goto2(BOX1), climbonbox(BOX1),
              climboffbox(BOX1), pushto(BOX1 LIGHTSWITCH1),
              climbonbox(BOX1), turnonlight(LIGHTSWITCH1)}
   PostSTRIPS solution:
             {goto2(BOX3), pushto(BOX3 LIGHTSWITCH1),
              climbonbox(BOX3), turnonlight(LIGHTSWITCH1),
              climboffbox(BOX3)}

2. Push three boxes together.
   Goal wff: NEXTTO(BOX1 BOX2) and NEXTTO(BOX2 BOX3)
   STRIPS solution:
             {goto2(BOX2), pushto(BOX2 BOX1),
              goto2(BOX3), pushto(BOX3 BOX2)}
   PostSTRIPS solution:
             {goto2(BOX2), pushto(BOX2 BOX1),
              goto2(BOX3), pushto(BOX3 BOX2)}

3. Go to a location in another room.
   Goal wff: ATROBOT(f)
   STRIPS solution:
             {goto2(DOOR1), gothrudoor(DOOR1 ROOM1 ROOM5),
              goto2(DOOR4), gothrudoor(DOOR4 ROOM5 ROOM4),
              goto1(f)}
   PostSTRIPS solution:
             {goto2(DOOR1), gothrudoor(DOOR1 ROOM1 ROOM5),
              goto2(DOOR4), gothrudoor(DOOR4 ROOM5 ROOM4),
              goto1(f)}
```

The first plan shows how the heuristics included in PostSTRIPS improve performance. The heuristic that shows its usefulness in this plan is: "Abandon a plan solution that takes you back to a world state that existed earlier in the planning process." PostSTRIPS considered climbing on the box early in the plan, as STRIPS did. When PostSTRIPS found that it had to climb off the box to satisfy another constraint, and when it found that this world state had existed earlier in the plan, it abandoned this solution. A solution that did not have this flaw was then discovered.

The first plan also shows the application of postconditions. The operator "turnonlight" was modified to include the postcondition "onfloor." Thus the PostSTRIPS solution includes the final step of "climboffbox." Requiring the robot to climb off the box when it is done turning off the light may not be desired in all cases. The example is here only to show one application of this added capability. STRIPS did not provide for a clean implementation of this capability.

The second plan demonstrates the action of the critic to provide a particular interpretation of a compound goal. PostSTRIPS, without the critic, accomplished each subgoal in turn without considering its affect on earlier subgoals. In this case, the initial plan was to go to box1, push box1 to box2, go to box2, and push box2 to box3. The critic examined this plan and rejected it since the first goal of having box1 next to box2 was no longer true. PostSTRIPS is able to continue its search for a solution and then provided the same solution that STRIPS provided.

The third plan was the last sample of STRIPS planning available and is provided for completeness. PostSTRIPS provides the same solution as STRIPS in this case also.

To demonstrate the utility of dividing the add list into effects and side effects, a comparison was made between execution time with and without this feature. The results are presented in Table 6.2.

TABLE 6.2

Program Execution Time

Task:	Single Addlist	Divided Addlist	Time Change	STRIPS
1. Turn on the lightswitch.	1.291	1.242	-3.8%	113.1
2. Push three boxes together.	.970	.951	-1.9%	66.0
3. Go to location in another room.	.715	.684	-4.4%	123.0

All times in seconds.

While the improvements in planning time are small, examination of the STRIPS operators (Chapter 3) will show that these simple operators do not have a large number of side effects. More complex operators, with a greater number of side effects should enjoy a greater speed up in planning time.

Also shown in Table 6.2 is the reported STRIPS execution time. This does not provide a fair comparison of algorithms because of significant hardware improvements. (Comparison of algorithms based on some other metric is also not reasonable because of the difference in the two programs' structure.) What the information does show is that in 1970 a person could perform such a planning task faster that a machine, in 1985 the situation is reversed.

In the flightline domain, PostSTRIPS is able to handle complex operators. Consider "ptrans-place:"

```
ptrans-place(m n):
Robot changes location of item m from n.

 Preconditions:
      ACCESSIBLE(ROBOT m) &
      [POSITIONABLE(ROBOT m) | MOVEABLE(ROBOT m)] &
      GRASPED(m) & [ROLLS(m) | LIFTED(ROBOT m)] &
      LOCINROOM(n x) & INROOM(ROBOT x)
 Postconditions:
      HANDEMPTY
 Delete list: AT(m $) NEXTTO(m $) NEXTTO($ m)
              +
              NEXTTO(ROBOT $) ATROBOT($)
    Add list: AT(m n)
              +
              ATROBOT(n) NEXTTO(ROBOT m).
```

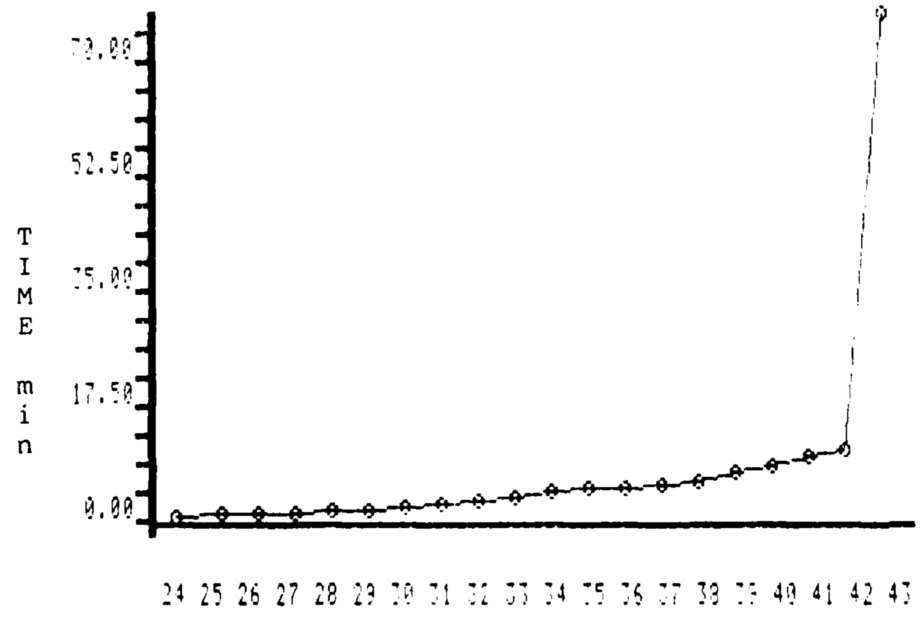
This operator is versatile enough to allow the robot to move items that are light enough to lift or that can be rolled. The divided effect and side effects keep the planner from trying to get to point n by moving something there.

A complete list of operators necessary to handle the F-15 and F-16 refueling tasks can be found in Appendix D. An initial world model to demonstrate these operators can be found in Appendix E. Using this initial world state, PostSTRIPS can take a list of wffs and produce a complete refueling plan. Appendix F contains the initial refueling scripts (the list of the wffs for each aircraft) and the complete refueling plans (obtained as output from the planner.) These complete plans form the input to the generalizer.

Generalizing

Initial attempts to use the first generalizer described in Chapter 5 failed. Execution time for the complex tasks was too great. While real-time operation was specifically excluded as a requirement in Chapter 1, the first generalizer did not provide a solution after 48 hours of computer time.

To investigate this, the generalizer was applied to partial planned scripts. Figure 6.1 is a graphic representation of number of script steps and time required to generalize them. Based on an extrapolation of the graph in Figure 6.1, it is apparent that two 70 or 80 step plans could not reach a solution in the time available to generate one.



Number of Task Steps

Figure 6.1 Early generalizer - execution time vs steps

The solution was to apply knowledge to the problem. The second generalizer described in Chapter 5 takes the unplanned refueling scripts and finds the commonalty of these plans. A small number of steps are common to the two plans. Each of these common steps is a (potential) dividing point for the planned refueling scripts. The generalizer can handle these smaller scripts, extract the common steps, and then splice the results together. Computation time for the generalization of the F-15 and F-16 planned scripts into a single script took under two minutes. This generalized plan can be found in Appendix G.

Summary, Conclusion, and Recommendations

Continued sponsorship of AI and robotic research by the Air Force is critical. Fifteen years ago, a computer program struggled for minutes to produce an answer to a simple planning problem that a man could produce as fast as he could speak the answer. Today that plan can be generated by a computer faster than a man can say it [Table 6.2]. Tremendous gains have been made. When state of the art reaches the point that robots can learn the same way man does, but faster, the AI credo "In the knowledge lies the power." will become "In the knowledge acquisition lies the power." The Air Force must remain on the leading edge of technology to exploit the leverage that these systems will offer.

To conclude this first look at automated knowledge acquisition, this chapter will summarize the work done, draw conclusions that can be drawn, and recommend directions for further research.

Summary

This research has involved both the robot and AI arenas. In the area of robotics, the operation of a robot on the flightline has been further examined. Due to the limited number of research vehicles available, most of the research has assumed single robot operation. If robots prove an economical tool, use of several will doubtless follow. Chapter 2 addressed problems and advantages in employing numerous semi-autonomous robots on the flightline. Another application, suggested in Chapter 4, is that of robots assisting men in performing tasks on the flightline. This blend of talents may prove the most valuable application while freeing up a number of humans to perform other tasks. Figure 7.1 summarizes the interactions of these players.

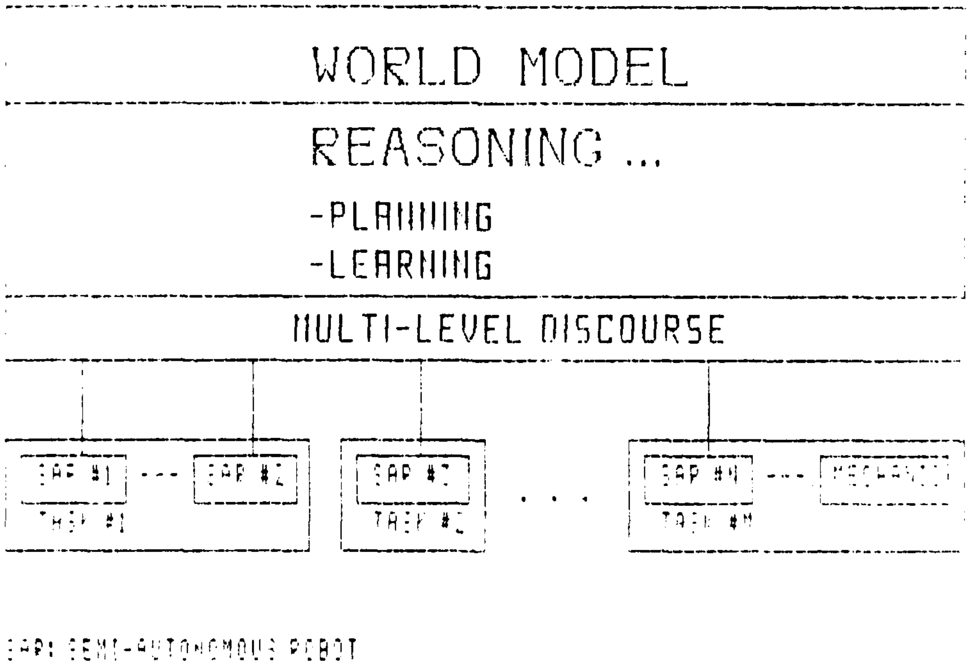


Figure 7.1 A robotic flightline model

Because of the value of the resources on the flightline, the greatest of which are the human resources, a call has been made to bring the safety issue into the open. Because of the complexity of robots, their actions in the presence of human labor may prove unpredictable and dangerous. Robots have already been responsible for the loss of human life and those doing robotic research can no longer ignore the safety issue. To simply wait and see what happens will doubtless continue to cost human lives.

In the area of AI, this research has investigated knowledge representation, knowledge transformation, planning, and generalizing. It has suggested an extension to the Conceptual Dependency representation scheme to include additional information that can resolve subtle differences in meaning while maintaining the generality that also proves useful.

This research has investigated the translation of actual maintenance Job Guides into a form that a machine might utilize. This involved the F-15 and F-16 refueling Job Guides, each written by different manufacturers in a slightly different style. A framework for a system to perform this task was presented in Chapter 5.

PostSTRIPS, a natural extension of the STRIPS planner was designed and implemented. It was tested in the original STRIPS domain and in the flightline robot domain.

The last area covered by this research was the area of learning. Learning was investigated in the form of generalizing specific instances of refueling scripts into an overall concept refueling.

Conclusions

The successful transformation of the F-15 and F-16 refueling Job Guides into a useful internal representation is due in part to the fact that it was done by hand. It proves that it can be done. Further investigation is required to determine how large a knowledge base is required to do the task by machine without constant restructuring or otherwise tweaking of the dictionary or other portions of the natural language interpreter. Authors of the Job Guides have gone to great lengths to make the guides understandable by all. But aircraft are complex machines, become more complex each day, and require a large vocabulary. Close inspection of the data shows that a moderately powerful natural language interpreter with minor extensions to its syntax handler should be able to accept the Job Guide form of input.

The ability of PostSTRIPS to plan two orders of magnitude faster than STRIPS should not have been a surprise. Advances in hardware speed are well documented. This particular gain in speed has serious implications since it tips the balance in favor of machines being able to handle planning problems faster than man. That AI techniques are really becoming a feasible tool for real time operation was unexpected and represents a significant data point in support of current interest in AI.

The ease with which the STRIPS operator was extended to include additional knowledge supports an intuitive feeling that the representation of primative capabilities by this scheme is useful. The further targeting of appropriate operators by separating operator effects from operator side effects shows promise of minimizing planning time. The ability to differentiate between subgoals that must be currently true and subgoals that can be made true proved useful. Finally, the ability to add post processing to operators rounds out the representation scheme.

Generalizing from specific instances to concepts is obviously a useful capability. People use it all the time. The average adult would have no problem refueling a car he has never seen before, given he can find the fuel cap. The ability to design a program to perform such a task is possible, as this research demonstrates. How to apply this capability to increase a robots usefulness is not obvious.

Recommendations

This research into machine learning, if nothing else, has confirmed the enormity of the task. Much remains to be discovered in this area. Because of the leverage automated knowledge acquisition would add to a system, research must be accelerated in this direction.

The effort must be divided to address all types of learning. While rote learning appears simplistic, much of the learning that a near term machine has to accomplish falls in this area. Industry is trying to find ways to transfer the program tapes generated by the "teach repeat" method on one machine to another. At the present, errors in the sensors prevent this. The amount of effort required to enter, debug, and edit information makes it costly to reproduce the present representation on other machines. A higher level representation of the task would solve this problem. The degree of understanding we have of this form of learning makes this area the most promising and productive in the near term.

Learning by being told holds the next most promising prospects in the near term. Operationalization of advice would allow a higher level of discourse between a man and machine and speed up thought transference. This will be required in some time critical tasks.

Learning from examples offers even greater gains but does not appear to offer them in near term prospects. One advantage would be to allow a greater information storage density as a system is able to dynamically restructure its knowledge when similarities are found.

Learning by analogy offers the greatest gains but also is the furthest from being realized. As an example, the dynamic restructuring that learning from examples offers could occur across domain boundaries. This would offer many times the information density of learning from examples.

The important point is that each type of learning offers features and advantages depending on how long a term one applies in long term planning. Because of the trade-offs between near term gains and total gains, and because of the not yet understood interaction of the four forms of learning, each must be investigated further.

Another point that requires further research is the area of natural language understanding. This paper "discovered" a short hand notation used in the Job Guide for representing the person responsible to accomplish a given task. An effort to exercise a more powerful natural language understanding system in the maintenance domain will probably show other unique symbology used in other flightline tasks. While partial systems have been built, more research needs to be done in determining whether the form produced by available interpreters is useful and extensible when applied to an existing domain.

As a third recommendation, since hardware for robotic research is limited, a simulation facility is needed to allow multiple robotic research projects to proceed. The system should include a motion capable graphics display and a straight forward way to add "actors" to the world model to provide a feeling for the interaction between components that cannot be gleaned by observing line after line of well formed formulas scrolling by on a screen. Such a project could be combined with a knowledge engineering tool such as IntelliCorp's KEE(tm) or Carnegie Group's Knowledge Craft(tm) to utilize available capabilities and avoid reconstructing these support portions of the system.

Finally, research on the topic of a flightline robot must continue. Further refining of the definition of such a robot is an appropriate single topic for a research effort. It may be premature to overly narrow the definition of the robot at this time. With aircraft that remain in the inventory for more than thirty years, some amount of advanced planning is needed to prevent the need to retrofit hardware so that the robot can work on it. It is time to start including discussions of cooperation between robots and robots and cooperation between robots and men. Safety issues must be brought to the foreground. A robot will be working on the flightline in less than thirty years.

Appendix A

F-15 and F-16 Refueling Job Guides

This appendix presents a comparison of F-15 and F-16 refueling Job Guides for "Refueling Without External Electrical Power." Note that prefatory information for the F-16 applies to all refueling operations while the F-15 Job Guide provides prefatory information for each type of refueling operation. Conceptually equivalent steps have been aligned.

TO 1F-15A-2-12JG-10-2	T.O. 1F-16A-2-12JG-00-1
INPUT CONDITIONS.	INPUT CONDITIONS
Applicability: All	Applicability: All
Required Conditions:	Required Conditions:
Aircraft safe for maintenance (05-00-01)	Aircraft safe for maintenance (JG10-30-01)
	Access door 3013 open (General Maintenance)
Personnel Recommended: Three	Personnel Recommended: Four
(A) Supervising operations	Technician A acts as refueling supervisor (between aircraft and fuel truck in full view of refueling operation, close to fire extinguisher positioned between aircraft and truck).
(B) Connecting and disconnecting equipment	Technician B verifies aircraft ground and bonding cables are properly connected, operates fuel nozzle at aircraft, and positions fire extinguisher close to nozzle during power-on refueling operations (aircraft ground cable).
(C) Operating fuel servicing equipment	Technician C operates fuel truck (fuel truck).
	Technician D operates aircraft controls, if power is applied (in cockpit).

Support Equipment:	Support Equipment:
Equipment, fuel servicing	Fuel Truck, Type AF/S32R-2 or equivalent.
Fire extinguisher, CO2, CB or Halon	Fire Extinguisher, CO2, 5Ø-pound or equivalent (two required when refueling with external power)
	(2) Generator Set, Type A/M32A-6ØA or equivalent
	(2) Headset-Microphone, Part No. H-133C/AIC (two)
	(2) Cordage Assembly, Part No. 5-62887-2
	(2) [1Ø] Wrench, Part No. 61Ø78-2 (three)

Support Data:

71-Ø3-Ø3 or 71-Ø4-Ø3 or 71-Ø5-Ø3

TO ØØ-25-172

TO 11A-1-33

Supplies (Consumables):

NOMENCLATURE	PART NO. (MFG CODE)	QTY
Turbine Fuel, Aviation	MILT5624GR JP4 (81349)	AR
or		
Turbine Fuel, Aviation	MILT5624GR JP5 (81349)	AR
or		
Turbine Fuel, Aviation	MILT83133 GRJP8 (81349)	AR

Supplies (Consumables):

NOMENCLATURE	SPECIFICATION	QTY
Jet Fuel JP-4	MIL-T-5624	AR
Jet Fuel JP-5 (Alternate)	MIL-T-5624	AR
Jet Fuel JP-8 (Alternate)	MIL-T-83133	AR

Safety Conditions:

---WARNING---

To prevent death or injury to personnel, precautions listed below must be complied with.

Aircraft and fuel servicing equipment are grounded.

Refueling is not done within 50 feet of aircraft with engine(s) operating.

Radar transmission is not directed toward refueling operation.

Refueling is not done within 110 feet of aircraft parked parallel with operating airborne radar equipment and airborne surveillance radar.

Refueling nozzle locking device and aircraft ground refueling receptacle are serviceable.

Wing vent/dump masts and centerline and inboard pylon vents are not obstructed.

If installed, emergency vent on external fuel tank is not obstructed.

Air flows from wing vent/dump masts and centerline and inboard pylon vents, if external fuel tanks are installed, during refueling.

Operation of jet fuel starter (JFS) is prohibited during normal refueling. Refueling with JFS operation may be required

Safety Conditions:

---WARNING---

Supervisor shall verify that each technician is familiar with Safety Conditions listed below and is briefed on position and responsibilities; otherwise, injury to personnel or damage to aircraft may occur.

Prolonged contact with fuel may cause skin irritation or disease. Saturated clothing shall be removed immediately and affected skin area shall be washed thoroughly.

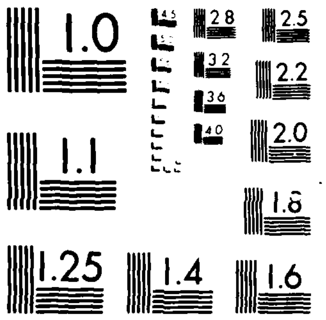
To avoid fire and explosive hazards, spilled fuel shall be cleaned up immediately.

If fuel splashes into eyes, wash eyes immediately; then seek medical services.

If any fuel spillage has occurred on surface of aircraft, area shall be checked to determine if fuel has impregnated insulating blankets or duct insulation. If evidence of impregnation exists, insulating blankets or duct insulation must be replaced prior to engine operation. Failure to observe this warning will create a potential fire hazard.

Do not service fuel in aircraft when an electrical storm is within a 3-mile radius of servicing area.

During fuel servicing, an area within a 50-foot radius of aircraft shall be restricted.


1.0
4.5
2.8
2.5
3.2
2.2
3.6
4.0
2.0
1.1
1.8
1.25
1.4
1.6

to satisfy quick-turn-around requirements, but is not to be operated with external fuel tanks installed.

One serviceable CO2, CB or Halon fire extinguisher is available.

Personnel shall be through thoroughly familiar with TO ØØ-25-172 for ground servicing and positioning of equipment.

Personnel and equipment are clear of landing gear doors at all times unless safety pins are installed.

All persons working in refueling shall dissipate static potential by gripping static ground wire connection at ground clamp and repeat often during refueling operation.

Aircraft explosives should be unloaded before refueling, however, if mission required, the local commander may approve refueling of explosive loaded aircraft. Refer to TO 11A-1-33.

Limit refueling pressure to 2Ø psi when aircraft vents fuel until fuel system had been tested per fuel level control valves operational checkout (28-24-Ø5).

Aircraft must be cleared for hot refueling (engine running) by a qualified weapons specialist, aircraft may be cleared for cold refueling (engine not running) by a weapons task

---CAUTION---

All support equipment not required for fuel servicing shall be moved a minimum of 5Ø feet from aircraft.

Engine shall not be started nor aircraft taxied or towed within 5Ø feet of fuel servicing activity, except as provided during fast turnaround.

Transmissions from attack radar shall not be permitted within a radius of 75 feet.

Transmissions from UHF and VHF radio shall be limited to emergency situations. Interphone communications shall be limited to transmissions necessary to accomplish servicing operation.

All movement of equipment in immediate vicinity of aircraft shall be accomplished under supervision of designated personnel to preclude possible damage to aircraft.

Position all support equipment required for fuel servicing at maximum distance from aircraft that hoses or cables will allow.

When servicing with external power, a minimum of two 5Ø-pound capacity carbon dioxide or 1Ø-gallon CBM fire extinguishers shall be used. One will be near generator set used for servicing and the other will be in immediate vicinity of servicing connection point.

When servicing without external power, a minimum of qualified individual.

one 50-pound capacity carbon dioxide or 10-gallon CBM fire extinguisher shall be used. It will be in immediate vicinity of servicing connection point.

Verify all stands and equipment are removed which might cause damage when aircraft shock struts compress or extend due to increased or decreased fuel load.

Equipment operator shall remain at the equipment and shall continually observe equipment and aircraft for any evidence of leakage during servicing operation. Operator shall be alert for any indication which might necessitate alteration or discontinuance of fuel servicing operation.

If fuel vapors considered by supervisor to be dangerous in any way are detected inside or outside aircraft, servicing operations shall cease immediately.

If fuel spills, supervisor shall order operations to cease immediately and shall notify base fire department. Special precautions shall be taken until safety hazard has been eliminated by fire department.

Before fuel servicing operation is started, all personnel not actively engaged in the operation shall be evacuated from area.

All personnel engaged in fuel servicing shall discharge electricity from their persons by touching a static ground cable or grounded object before each operation.

All personnel engaged in fuel servicing operations must be familiar with complete fuel servicing procedures. Each person must be instructed as to specific duties prior to starting any fuel servicing operation.

Verify that main gear wheels are not chocked tightly since an increase in aircraft weight will result in chocks being locked under main gear wheels when aircraft is being serviced.

During fuel servicing, verify fuel vent remains free of obstructions.

---NOTE---

If problems involving the fuel venting system are encountered anytime during refueling. See fault code (28-23-XS).

Other Recommendations:

Item T.O. 00-25-172 to provide instructions for aircraft grounding sequence.

Checklist Requirement: This function shall be followed in step-by-step checklist sequence to prevent damage to equipment or injury to personnel.

PROCEDURE.	REFUELING WITHOUT EXTERNAL POWER.
1. (A) If aircraft is installed on IAC (Industrial Acoustics Company), Jetway Noise Suppressor System, or EECORP (Environmental Elements Corporation) do the below:	
1.1. (A) Left and right PAIM (primary air intake mufflers) are retracted and secured (71-03-03) IAC, (71-04-03) Jetway or (71-05-03) EECORP.	
1.2. (A) Noise suppressor system main electrical power switch is off.	
-----NOTE----- A fire fighting apparatus will be used at noise suppressor systems.	
1.3. (A) Mobile fire fighting apparatus (fire truck) is standing by. Size will be determined by fire chief, but will not be less than P13 capability.	
	-----NOTE----- Aircraft shall be positioned on a level ramp, if possible.
2. (B) Position fire extinguisher and make sure aircraft and fueling equipment are grounded.	1. (B) Ground aircraft (T.O. 00-25-172).
	2. (C) Position fuel truck upwind.
	3. (B) Connect truck ground cable to same ground as aircraft.
	4. (B) Connect static bond cable between truck and

aircraft.

5. (A) Position one fire extinguisher between aircraft and truck.

-----NOTE-----

If external tanks are installed, verify tanks are depressurized by depressing negative pressure relief valve poppet in external tank.

3. (B) Open door 29.

6. (B) Verify fuel vent is fr of obstructions.

7. (B) Visually inspect fuel nozzle for damage and serviceability.

8. (B) Unreel fuel servicing hose.

4. (B) Remove ground refueling receptacle cap.

9. (B) Remove cap from aircraft refueling adapter.

---WARNING---

To prevent fuel spillage, resulting in possible fire and/or explosion, make sure refueling nozzle is locked into position.

10. (B) Visually inspect adapter for damage and serviceability.

5. (B) Insert refueling nozzle in ground receptacle by pushing up and turning clockwise until nozzle resists turning.

11. (B) Connect nozzle to refuel adapter.

6. (B) Lock nozzle into position by setting manual shutoff lever in full open position.

12. (B) Open nozzle valve.

---CAUTION---

If external tanks are installed and require fuel, verify tank drains are fully closed. Failure to comply will result in fuel spillage, creating a potential fire hazard.

-----NOTE-----

If external tanks require fuel, omit step 13.

13. [10] (B) Using wrenches, close three external fuel tank manual refueling valves.

7. (B) Observe refueling nozzle engagement by a counterclockwise tug on nozzle handles. Determine refueling hose quick disconnect fitting engagement by pulling on hose and visual inspection.

---CAUTION---

Prior to application of fuel pressure, verify nozzle cannot be rotated and disconnected. Failure to observe this caution may allow nozzle to become disconnected from refuel adapter with valve remaining in open position during fuel servicing, creating a potential fire hazard.

Do not exceed 60 psi nozzle pressure.

8. (B) Stand by fire extinguisher.

---CAUTION---

To prevent possible damage to fuel system, fuel pressure from servicing equipment shall not exceed 55 psig.

9. (A and C) Make sure aircraft is receiving correct fuel (JP-4, JP-5 or JP-8), then start fuel servicing equipment and start refueling.

-----NOTE-----

Do steps 10 and 11 within 1 minute of refueling operation.

10. (A) Make sure air flows from wing vent/dump masts and centerline and inboard pylons if external fuel tanks are receiving fuel.

11. (A and B) If air flows from only one wing vent/dump mast, restrict airflow with flat of hand and observe opposite side to make sure air flows from wing vent/dump mast.

12. If fuel flow does not stop automatically fuel spillage will result, do the below immediately:

12.1. (C) Stop fuel service operation immediately and turn off fuel servicing equipment.

12.2. (B) Disconnect refueling nozzle from ground

14. (C) Start fuel pump to begin servicing of fuel.

-----NOTE-----

If aircraft is not laterally level, the high wing may not completely fill. This can be corrected by manually shaking the wing to allow air to flow inboard before closing nozzle valve.

refueling receptacle.	
12.3. (A) Contact fire department and take required action to make area safe before moving aircraft or ground support equipment.	
13. If fuel flow does not automatically stop, malfunction exists, refer to Ground Refueling and Defueling Malfunction, Fault Code 2824B.	
14. When refueling for fuel quantity system calibration, maintain fuel pressure for another 5 minutes to make sure refueling is correct.	
15. (C) When fuel flow automatically stops, turn off fuel servicing equipment.	15. (B) When fuel flow stops, close nozzle valve.
	16. (C) Stop fuel pump.
	-----NOTE-----
	17. [1Ø] (B) Using wrenches, open three external fuel tank manual refueling valves.
16. (B) Disconnect refueling nozzle from ground refueling receptacle.	18. (B) Disconnect nozzle
17. (B) Install ground refueling cap.	19. (B) Install cap on aircraft refueling adapter.
18. (B) Do foreign object inspection of area and close door 29.	
	2Ø. (B) Reel up and stow fuel servicing hose.
	21. (B) Remove static bond cable.
	22. (B) Remove truck ground cable.

Appendix B

F-15 Job Guide Transformation Steps

This appendix shows how the F-15 Job Guide can be written as conceptual Dependencies and how those can be expressed as STRIPS like Operators. Operationalization has been done in converting CDs to extended CDs. This world knowledge is necessary to perform the task. Extended CDs have the ability to express these nuances that are needed to select the most appropriate STRIPS Operator.

Job Guide:

1. (A) If aircraft is installed on IAC (Industrial Acoustics Company), Jetway Noise Suppressor System, or EECORP (Environmental Elements Corporation) do the below:

1.1. (A) Left and right PAIM (primary air intake mufflers) are retracted and secured (71-03-03) IAC, (71-04-03) Jetway or (71-05-03) EECORP.

1.2. (A) Noise suppressor system main electrical power switch is off.

-----NOTE-----

A fire fighting apparatus will be used at noise suppressor systems.

1.3. (A) Mobile fire fighting apparatus (fire truck) is standing by. Size will be determined by fire chief, but will not be less than P13 capability.

Conceptual Dependency form:

((TEST . . .

Note: It should be apparent from the above that this is a complex situation. Rather than expend resources on this special (rare) case, the author has chosen to concentrate on the more common and more comprehensible situation in which a noise suppression testbed is not involved.

STRIPS form:

case-IAC-EECORP(aircraft(F-15))

Note: This operator will always fail if aircraft is connected to a noise suppression testbed.

Job Guide:

2. (B) Position fire extinguisher and make sure aircraft and fueling equipment are grounded.

Conceptual Dependency form:

```
(AND (PTRANS (VERB   ((SPECIFIC position)))
             (ACTOR  man-B)
             (OBJECT fire-extinguisher)
             (SOURCE nil)
             (DESTIN nil))
     (ACHIEVE-PHYSICAL-STATE
             (VERB   ((SPECIFIC ground)
                      (STATE electric-connect)))
             (ACTOR  man-B)
             (OBJECT (and aircraft fueling-equipment))
             (DESTIN nil)))
```

CD Operationalized:

```
(PTRANS (VERB   ((SPECIFIC position)))
        (ACTOR  man-B)
        (OBJECT fire-extinguisher)
        (SOURCE nil)
        (DESTIN (between aircraft(F-15) fuel-source)))
(ACHIEVE-PHYSICAL-STATE
        (VERB   ((SPECIFIC ground)
                 (STATE electric-connect)))
        (ACTOR  man-B)
        (OBJECT ground-point(F-15))
        (DESTIN nearest-earth-ground-point)
        (ORGAN  ground-wire))
(ACHIEVE-PHYSICAL-STATE
        (VERB   ((SPECIFIC ground)
                 (STATE electric-connect)))
        (ACTOR  man-B)
        (OBJECT fuel-source ((SPECIFIC fueling-equipment)))
        (DESTIN aircraft-earth-ground-point(F-15))
        (ORGAN  ground-wire))
```

STRIPS form:

```
ptrans-place(fire-extinguisher
           between(aircraft(F-15) fuel-source))
ground-aircraft(aircraft(F-15))
ground-fuel-source(fuel-source
                 aircraft-earth-ground-point(F-15))
```

Job Guide:

3. (B) Open door 29.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC open)))
      (ACTOR  man-B)
      (OBJECT door(29))
      (SOURCE nil)
      (DESTIN nil))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC open)))
      (ACTOR  man-B)
      (OBJECT door(29))
      (SOURCE nil)
      (DESTIN open)
      (ORGAN  hand))
```

STRIPS form:

```
open-access-door(door(29))
```

Job Guide:

4. (B) Remove ground refueling receptacle cap.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC remove)))
      (ACTOR  man-B)
      (OBJECT refueling-receptacle-cap)
      (SOURCE nil)
      (DESTIN nil))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC remove)))
      (ACTOR  man-B)
      (OBJECT fuel-cap(F-15)
              ((SPECIFIC refueling-receptacle-cap)))
      (SOURCE SPR(F-15))
      (DESTIN stowage-point)
      (ORGAN  hand))
```

STRIPS form:

```
extract(fuel-cap(F-15) SPR(F-15))
move(fuel-cap(F-15) stowage-point)
```

NOTE: SPR is a common abbreviation for single point refueling.

Job Guide:

---WARNING---

To prevent fuel spillage, resulting in possible fire and/or explosion, make sure refueling nozzle is locked into position.

5. (B) Insert refueling nozzle in ground receptacle by pushing up and turning clockwise until nozzle resists turning.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC insert)))
      (ACTOR  man-B)
      (OBJECT refueling-nozzle)
      (SOURCE nil)
      (DESTIN ground-receptacle))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC insert)))
      (ACTOR  man-B)
      (OBJECT fuel-nozzle ((SPECIFIC refueling-nozzle)))
      (SOURCE nil)
      (DESTIN SPR(F-15) ((SPECIFIC ground-receptacle)))
      (ORGAN  hand))
```

STRIPS form:

```
fuel-nozzle-connected(SPR(F-15))
```

Job Guide:

6. (B) Lock nozzle into position by setting manual shutoff lever in full open position.

Conceptual Dependency form:

```
(ACHIEVE-
PHYSICAL-STATE (VERB ((SPECIFIC lock)
                      (STATE lock-in-position)))
               (BY    (MOVE (VERB   ((SPECIFIC setting)))
                      (ACTOR  man-B)
                      (OBJECT manual-shutoff-lever)
                      (SOURCE nil)
                      (DESTIN full-open-position))))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC setting)))
      (ACTOR  man-B)
      (OBJECT fuel-nozzle-valve
              ((SPECIFIC manual-shutoff-lever)))
      (SOURCE nil)
      (DESTIN open ((SPECIFIC full-open-position)))
      (ORGAN  hand))
```

STRIPS form:

```
control(fuel-nozzle-valve open)
```

Job Guide:
7. (B) Observe refueling nozzle engagement by a counterclockwise tug on nozzle handles. Determine refueling hose quick disconnect fitting engagement by pulling on hose and visual inspection.

Conceptual Dependency form:

```
(ATTEND (VERB ((SPECIFIC  observe)))
        (NOTE refueling-nozzle-engagement)
        (BY
   (APPLY-FORCE (VERB ((SPECIFIC tug)
                      (DIRECTION counterclockwise)))
                (ACTOR  man-B)
                (OBJECT nozzle-handles))
(ATTEND (VERB   ((SPECIFIC determine)))
        (NOTE   engagement)
        (BY
   (AND (APPLY-FORCE (VERB   ((SPECIFIC  pull)
                              (DIRECTION downward)))
                     (ACTOR  man-B)
                     (OBJECT hose-quick-disconnect-fitting))
        (ATTEND (VERB   ((SPECIFIC        inspection)
                          (IMPLIED-DEFALUT vision)))
                (ACTOR  man-B)
                (OBJECT hose-quick-disconnect-fitting)
                (NOTE   engagement))))
```

CD Operationalized:

```
(APPLY-FORCE (VERB   ((SPECIFIC  tug)
                      (AMOUNT    10-ft-lbs)
                      (DIRECTION counterclockwise)))
             (ACTOR  man-B)
             (OBJECT nozzle-handles)
             (ORGAN  hand))
(TEST (VERB   ((SPECIFIC nil)))
      (ACTOR  man-B)
      (OBJECT nozzle-handles)
      (COND   (equal source destin)))
(APPLY-FORCE (VERB   ((SPECIFIC  pull)
                      (AMOUNT    10-ft-lbs)
                      (DIRECTION downward)))
             (ACTOR  man-B)
             (OBJECT hose-quick-disconnect-fitting)
             (ORGAN  hand))
(TEST (VERB   ((SPECIFIC nil)))
      (ACTOR  man-B)
      (OBJECT hose-quick-disconnect-fitting)
      (COND   (equal source destin)))
(ATTEND (VERB   ((SPECIFIC        inspection)
                  (IMPLIED-DEFAULT vision)))
```

```
       (ACTOR man-B)
       (OBJECT hose-quick-disconnect-fitting)
       (NOTE   engagement))
```

STRIPS Operator:

```
attend(fuel-nozzle engagement)
attend(quick-disconnect engagement)
```

NOTE: For planning purposes these are assumed to be true. Failure would be handled during plan execution.

Job Guide:

8. (B) Stand by fire extinguisher.

Conceptual Dependency form:

```
(ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC stand)))
                        (ACTOR  man-B)
                        (OBJECT man-B)
                        (SOURCE nil)
                        (DESTIN (prox fire-extinguisher)))
```

CD Operationalized:

```
(PTRANS (VERB   ((SPECIFIC nil)))
        (ACTOR  man-B)
        (OBJECT man-B)
        (SOURCE nil)
        (DESTIN fire-extinguisher))
```

STRIPS form:

```
goto-item(fire-extinguisher)
```

Job Guide:

---CAUTION---

To prevent possible damage to fuel system, fuel pressure from servicing equipment shall not exceed 55 psig.

9. (A and C) Make sure aircraft is receiving correct fuel (JP-4, JP-5 or JP-8), then start fuel servicing equipment and start refueling.

Conceptual Dependency form:

```
(ATTEND (VERB   ((SPECIFIC make-sure)
                 (IMPLIED-ORGAN nil)))
        (ACTOR  (and man-A man-C))
        (OBJECT fuel-type)
        (NOTE   (or (equal type JP-4)
                    (equal type JP-5)
                    (equal type JP-8))))
(AND (ACHIEVE-
PHYSICAL-STATE (VERB   ((SPECIFIC start)
                         (STATE operating)))
               (ACTOR  (and man-A man-C))
               (OBJECT fuel-source
                     ((SPECIFIC fuel-servicing-equipment)))
               (DESTIN nil))
(ACHIEVE-
PHYSICAL-STATE (VERB   ((SPECIFIC start)
                         (STATE refueling)))
               (ACTOR  (and man-A man-C))))
```

CD Operationalized:

```
(ATTEND (VERB   ((SPECIFIC make-sure)
                 (IMPLIED-ORGAN nil)))
        (ACTOR  (and man-A man-C))
        (OBJECT fuel-type)
        (NOTE   (or (equal type JP-4)
                    (equal type JP-5)
                    (equal type JP-8))))
(ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC start)
                                  (STATE operating)))
                        (ACTOR  man-C)
                        (OBJECT fuel-source)
                        (DESTIN on))
```

STRIPS form:

```
attend(fuel-type (or (equal type JP-4)
                     (equal type JP-5)
                     (equal type JP-8)))
control(fuel-source on)
```

Job Guide:

-----NOTE-----

Do steps 1Ø and 11 within 1 minute of refueling operation.

1Ø. (A) Make sure air flows from wing vent/dump masts and centerline and inboard pylons if external fuel tanks are receiving fuel.

Conceptual Dependency form:

```
(ATTEND (VERB   ((SPECIFIC make-sure)
                 (IMPLIED-ORGAN nil)))
        (ACTOR  man-A)
        (OBJECT (and (wing-vent/dump-masts)
                     (and installed (and centerline-pylon
                                         inboard-pylon))))
        (ORGAN  nil)
        (NOTE   air-flow))
```

CD Operationalized:

```
(ATTEND (VERB   ((SPECIFIC make-sure)
                 (IMPLIED-ORGAN nil)))
        (ACTOR  man-A)
        (OBJECT (and (wing-vent/dump-masts)
                     (and installed (and centerline-pylon
                                         inboard-pylon))))
        (ORGAN  hand)
        (NOTE   air-flow))
```

STRIPS form:

```
attend((and (wing-vent/dump-masts)
            (and installed (and centerline-pylon
                                inboard-pylon))) airflow)
```

NOTE: A special sensor might be necessary if the robot is to actually perform this task.

Job Guide:

11. (A and B) If air flows from only one wing vent/dump mast, restrict airflow with flat of hand and observe opposite side to make sure air flows from wing vent/dump mast.

Conceptual Dependency form:

```
(TEST (VERB ((SPECIFIC implied)
             (CONDITION air-flow-from-only-one-mast)))
             (ACTION
   (ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC restrict)
                                    (STATE no-air-flow)))
                           (ACTOR  (and man-A man-B))
                           (OBJECT (working-mast))
                           (DESTIN nil)
                           (ORGAN  hand))
   (ATTEND (VERB   ((SPECIFIC make-sure)
                    (IMPLIED-DEFAULT nil)))
           (ACTOR  (and man-A man-C)
           (OBJECT opposite-mast)
           (ORGAN  nil)
           (NOTE   air-flow)))
```

Conceptual Dependency form:

```
(TEST (VERB ((SPECIFIC implied)
             (CONDITION air-flow-from-only-one-mast)))
             (ACTION
   (ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC restrict)
                                    (STATE no-air-flow)))
                           (ACTOR  man-A)
                           (OBJECT (working-mast(F-15)))
                           (DESTIN nil)
                           (ORGAN  hand))
   (ATTEND (VERB   ((SPECIFIC make-sure)
                    (IMPLIED-DEFAULT nil)))
           (ACTOR  man-B)
           (OBJECT opposite-mast(F-15))
           (ORGAN  nil)
           (NOTE   air-flow)))
```

STRIPS form:

```
case-airflow-difficulty(fuel-vent(F-15))
```

NOTE Again special sensors are needed. This is beyond the purpose of this paper.

Job Guide:

12. If fuel flow does not stop automatically fuel spillage will result, do the below immediately:

12.1. (C) Stop fuel service operation immediately and turn off fuel servicing equipment.

12.2. (B) Disconnect refueling nozzle from ground refueling receptacle.

12.3. (A) Contact fire department and take required action to make area safe before moving aircraft or ground support equipment.

Conceptual Dependency form:

```
(TEST (VERB   ((SPECIFIC implied)
               (CONDITION fuel-spill)
               (ACTION
  (ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC stop)
                                    (STATE operations)))
                          (ACTOR  man-C))
  (ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC turn)
                                    (STATE off)))
                          (ACTOR  man-C)
                          (OBJECT fuel-servicing-equipment))
  (MOVE (VERB   ((SPECIFIC disconnect)))
        (ACTOR  man-B)
        (OBJECT fuel-nozzle ((SPECIFIC refueling-nozzle)))
        (SOURCE SPR(F-15))
                ((SPECIFIC ground-refueling-receptacle)))
        (DESTIN nil))
  (ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC contact)
                                    (STATE contact)))
                          (ACTOR  man-A)
                          (OBJECT fire-department))
  (ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC make)
                                    (STATE area-safe)))
                          (ACTOR  man-A)))
```

STRIPS form:

```
case-fuel-spill(AIRCRAFT(F-15))
```

NOTE: This would be implemented as a separate script. The condition would be assumed not true for planning purposes.

Job Guide:

13. If fuel flow does not automatically stop, malfunction exists, refer to Ground Refueling and Defueling Malfunction, Fault Code 2824B.

Conceptual Dependency form:

```
(TEST (VERB   ((SPECIFIC implied)
                (CONDITION fuel-spill)
                (ACTION
   (ATTEND (VERB   ((SPECIFIC refer)))
           (ACTOR  nil)
           (OBJECT maintenance-manual
                   ((SPECIFIC FaultCode-2824B)))))
```

STRIPS form:

```
case-fuel-spill-repair(AIRCRAFT(F-15))
```

Job Guide:

14. When refueling for fuel quantity system calibration, maintain fuel pressure for another 5 minutes to make sure refueling is correct.

Conceptual Dependency form:

```
(TEST (VERB   ((SPECIFIC implied)
                (CONDITION fuel-system-calibration)
                (ACTION
   (APPLY-PRESSURE (VERB   ((SPECIFIC maintain-pressure)))
                   (ACTOR  nil)
                   (OBJECT fuel-pressure)
                   (DESTIN nil))
                (DURATION five-minutes))
```

STRIPS form:

```
case-fuel-system-calibration-check(AIRCRAFT(F-15))
```

NOTE: Another instance of a procedure which will not be common and initially would not be performed by the robot. STRIPS should assume condition is not true for planning and handle the event at execution time if it should occur.

Job Guide:

15. (C) When fuel flow automatically stops, turn off fuel servicing equipment.

Conceptual Dependency form:

```
(ATTEND (VERB   ((SPECIFIC implied)))
        (ACTOR  man-C)
        (OBJECT fuel-flow-indicator)
        (NOTE   stop-increasing))
(MOVE (VERB   ((SPECIFIC turn)))
      (ACTOR  man-C)
      (OBJECT fuel-servicing-equipment)
      (SOURCE nil)
      (DESTIN off))
```

CD Operationalized:

```
(ATTEND (VERB   ((SPECIFIC implied)))
        (ACTOR  man-C)
        (OBJECT fuel-flow-indicator)
        (ORGAN  sight)
        (NOTE   stop-increasing))
(MOVE (VERB   ((SPECIFIC turn)))
      (ACTOR  man-C)
      (OBJECT fuel-source
              ((SPECIFIC servicing-equipment)))
      (SOURCE nil)
      (DESTIN off)
      (ORGAN  hand))
```

STRIPS form:

```
attend(fuel-flow-indicator stop-increasing)
control(fuel-source off)
```

Job Guide:

16. (B) Disconnect refueling nozzle from ground refueling receptacle.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC disconnect)))
      (ACTOR  man-B)
      (OBJECT refueling-nozzle)
      (SOURCE SPR ((SPECIFIC refueling-receptacle)))
      (DESTIN nil))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC disconnect)))
      (ACTOR  man-B)
      (OBJECT fuel-nozzle ((SPECIFIC refueling-nozzle)))
      (SOURCE SPR(F-15) ((SPECIFIC refueling-receptacle)))
      (DESTIN ground)
      (ORGAN  hand))
```

STRIPS form:

```
fuel-nozzle-disconnected(SPR(F-15))
```

Job Guide:

17. (B) Install ground refueling cap.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC install)))
      (ACTOR  man-B)
      (OBJECT ground-refueling-cap)
      (SOURCE nil)
      (DESTIN refueling-receptacle))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC install)))
      (ACTOR  man-B)
      (OBJECT fuel-cap(F-15)
              ((SPECIFIC ground-refueling-cap)))
      (SOURCE nil)
      (DESTIN SPR(F-15) ((SPECIFIC refueling-receptacle)))
      (ORGAN  hand))
```

STRIPS form:

```
twist(fuel-cap(F-15) SPR(F-15))
```

Job Guide:

18. (B) Do foreign object inspection of area and close door 29.

Conceptual Dependency form:

```
(AND (ATTEND (VERB   ((SPECIFIC inspection)))
             (ACTOR  man-B)
             (OBJECT area)
             (ORGAN  nil)
             (NOTE   foreign-objects))
     (MOVE (VERB   ((SPECIFIC close)))
           (ACTOR  man-B)
           (OBJECT door(29))
           (SOURCE nil)
           (DESTIN nil)))
```

CD Operationalized:

```
(ATTEND (VERB   ((SPECIFIC inspection)))
        (ACTOR  man-B)
        (OBJECT area)
        (ORGAN  nil)
        (NOTE   foreign-objects))
(MOVE (VERB   ((SPECIFIC close)))
      (ACTOR  man-B)
      (OBJECT door(29))
      (SOURCE nil)
      (DESTIN close)
      (ORGAN  hand))
```

STRIPS form:

```
attend(area (foreign-objects))
close-access-door(door(29))
```

Appendix C

F-16 Job Guide Transformation Steps

This appendix shows how the F-16 Job Guide can be written as Conceptual Dependencies (CDs) and how those can be expressed as STRIPS like Operators. Operationalization has been done in converting CDs to extended CDs. This world knowledge is necessary to perform the task. Extended CDs have the ability to express these nuances that are needed to select the most appropriate STRIPS Operator.

Job Guide entry:

1. (B) Ground aircraft (T.O. 00-25-172).

Conceptual Dependency form:

```
(ACHIEVE-PHYSICAL-STATE (VERB ((SPECIFIC ground)
                               (STATE  electric-connect)))
                        (ACTOR  man-B)
                        (OBJECT aircraft))
```

CD Operationalized:

```
(ACHIEVE-PHYSICAL-STATE (VERB    ((SPECIFIC ground)
                                   (STATE electric-connect)))
                        (ACTOR  man-B)
                        (OBJECT ground-point(F-16)
                                 ((SPECIFIC aircraft)))
                        (DESTIN nearest-earth-ground-point)
                        (ORGAN  ground-wire))
```

STRIPS form:

```
ground-aircraft(ground-point(F-16))
```

Job Guide entry:

2. (C) Position fuel truck upwind.

Conceptual Dependency form:

```
(PTRANS (VERB   ((SPECIFIC position)))
        (ACTOR  man-C)
        (OBJECT fuel-truck)
        (SOURCE nil)
        (DESTIN upwind))
```

CD Operationalized:

```
(PTRANS (VERB   ((SPECIFIC position)))
        (ACTOR  man-C)
        (OBJECT fuel-source ((SPECIFIC fuel-truck)
                             (TYPE AF/S32R-2))
        (SOURCE nil)
        (DESTIN upwind(aircraft(F-16))))
```

STRIPS form:

```
ptrans-place(fuel-source upwind(aircraft(F-16)))
```

Function upwind, given an aircraft, takes its location, the global knowledge of wind direction, the world map, and the global knowledge of length of a fuel truck hose and returns an appropriate location.

Job Guide entry:

3. (B) Connect truck ground cable to same ground as aircraft.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC connect)))
      (ACTOR  man-B)
      (OBJECT truck-ground-cable)
      (SOURCE nil)
      (DESTIN same-ground-as-aircraft))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC connect)))
      (ACTOR  man-B)
      (OBJECT fuel-source-ground-cable-clamp
              ((SPECIFIC truck-ground-cable)))
      (SOURCE stowage-point)
      (DESTIN aircraft-earth-ground-point(F-16)
              ((SPECIFIC same-ground-as-aircraft)))
      (ORGAN  hand))
```

STRIPS form:

```
attach(fuel-source-ground-cable-clamp
       AIRCRAFT-EARTH-GROUND-POINT(F-16))
```

Job Guide entry:

4. (B) Connect static bond cable between truck and aircraft.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC connect)))
      (ACTOR  man-B)
      (OBJECT static-bond-cable)
      (SOURCE truck)
      (DESTIN aircraft))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC connect)))
      (ACTOR  man-B)
      (OBJECT static-bond-cable-plug
              ((SPECIFIC static-bond-cable)))
      (SOURCE stowage-point)
      (DESTIN ground-point(F-16)
              ((SPECIFIC aircraft)))
      (ORGAN  hand))
```

STRIPS form:

```
insert(static-bond-cable-plug ground-point(F-16))
```

Job Guide entry:

5. (A) Position one fire extinguisher between aircraft and truck.

Conceptual Dependency form:

```
(PTRANS (VERB   ((SPECIFIC position)))
        (ACTOR  man-A)
        (OBJECT fire-extinguisher)
        (SOURCE nil)
        (DESTIN between(aircraft truck)))
```

CD Operationalized:

```
(PTRANS (VERB   ((SPECIFIC position)))
        (ACTOR  man-A)
        (OBJECT fire-extinguisher)
        (SOURCE nil)
        (DESTIN between(aircraft(F-16) fuel-source))
```

STRIPS form:

```
ptrans-place(fire-extinguisher
             between(aircraft(F-16) fuel-source))
```

Function "between" takes the location of two items and finds the midpoint between them.

Job Guide entry:

6. (B) Verify fuel vent is free of obstructions.

Conceptual Dependency form:

```
(ATTEND (VERB    ((SPECIFIC verify)
                   (IMPLIED-DEFAULT nil)))
        (ACTOR  man-B)
        (OBJECT fuel-vent)
        (ORGAN  nil)
        (NOTE   obstruction-free))
```

CD Operationalized:

```
(ATTEND (VERB    ((SPECIFIC verify)
                   (IMPLIED-DEFAULT nil)))
        (ACTOR  man-B)
        (OBJECT fuel-vent)
        (ORGAN  sight)
        (NOTE   obstruction-free))
```

STRIPS form:

```
attend(fuel-vent obstruction-free)
```

Job Guide entry:

7. (B) Visually inspect fuel nozzle for damage and serviceability.

Conceptual Dependency form:

```
(ATTEND (VERB   ((SPECIFIC inspect)
                  (IMPLIED-DEFAULT nil)
                  (TYPE visual))
        (ACTOR  man-B)
        (OBJECT fuel-nozzle)
        (ORGAN  nil)
        (NOTE   (AND damage serviceability)))
```

CD Operationalized:

```
(ATTEND (VERB   ((SPECIFIC inspect)
                  (IMPLIED-DEFAULT nil)
                  (TYPE visual))
        (ACTOR  man-B)
        (OBJECT fuel-nozzle)
        (ORGAN  sight)
        (NOTE   (AND (NOT damage) serviceability)))
```

Note: Here world knowledge is required to know that damage is not a good thing, but serviceability is and if either equates to a bad state, operations must be suspended until problem is corrected.

STRIPS form:

```
attend(fuel-nozzle (AND (NOT damage) servicable))
```

Job Guide entry:

8. (B) Unreel fuel servicing hose.

Conceptual Dependency form:

```
(ACHIEVE-
PHYSICAL-STATE (VERB ((SPECIFIC unreel)
                      (STATE not-wrapped-around-reel)))
               (ACTOR  man-B)
               (OBJECT fuel-servicing-hose)
               (SOURCE nil)
               (DESTIN nil))
```

CD Operationalized:

```
(ACHIEVE-
PHYSICAL-STATE (VERB ((SPECIFIC unreel)
                      (STATE not-wrapped-around-reel)))
               (ACTOR  man-B)
               (OBJECT fuel-hose
                       ((SPECIFIC fuel-servicing-hose)))
               (SOURCE fuel-truck-reel)
               (DESTIN floor)
               (ORGAN  hands))
```

STRIPS form:

```
unreel(fuel-hose).
```

Job Guide entry:

9. (B) Remove cap from aircraft fuel adapter.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC remove))
      (ACTOR  man-B)
      (OBJECT cap)
      (SOURCE aircraft-fuel-adapter)
      (DESTIN nil))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC remove))
      (ACTOR  man-B)
      (OBJECT fuel-cap(F-16) ((SPECIFIC cap)))
      (SOURCE SPR(F-16)
              ((SPECIFIC aircraft-fuel-adapter)))
      (DESTIN stowage-point)
      (ORGAN  hand))
```

STRIPS Operator:

```
extract(fuel-cap(F-16) SPR(F-16))
move(fuel-cap(F-16) stowage-point)
```

NOTE: SPR is a common abbreviation for single point refueling.

Job Guide entry:

10. (B) Visually inspect adapter for damage and serviceability.

Conceptual Dependency form:

```
(ATTEND (VERB    ((SPECIFIC inspect)
                  (IMPLIED-DEFAULT nil)
                  (TYPE visual))
        (ACTOR   man-B)
        (OBJECT  adapter)
        (ORGAN   nil)
        (NOTE    (AND damage serviceability)))
```

CD Operationalized:

```
(ATTEND (VERB    ((SPECIFIC inspect)
                  (IMPLIED-DEFAULT nil)
                  (TYPE visual))
        (ACTOR   man-B)
        (OBJECT  SPR(F-16)
                 ((SPECIFIC adapter)))
        (ORGAN   sight)
        (NOTE    (AND (NOT damage) serviceability)))
```

Note: Here world knowledge is required to know that damage is not a good thing, but serviceability is and if either equates to a bad state, operations must be suspended until problem is corrected.

STRIPS form:

```
attend(SPR(F-16) (AND (NOT damage) servicable))
```

Job Guide entry:

11. (B) Connect nozzle to adapter.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC connect)))
      (ACTOR  man-B)
      (OBJECT nozzle)
      (SOURCE nil)
      (DESTIN adapter))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC connect)))
      (ACTOR  man-B)
      (OBJECT fuel-nozzle ((SPECIFIC nozzle)))
      (SOURCE floor)
      (DESTIN SPR(F-16) ((SPECIFIC adapter)))
      (ORGAN  hand))
```

STRIPS form:

```
fuel-nozzle-connected(SPR(F-16))
```

12. (B) Open nozzle valve.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC open)))
      (ACTOR  man-B)
      (OBJECT nozzle-valve)
      (SOURCE nil)
      (DESTIN nil))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC open)))
      (ACTOR  man-B)
      (OBJECT fuel-nozzle-valve  ((SPECIFIC nozzle-valve)))
      (SOURCE nil)
      (DESTIN open)
      (ORGAN  hand))
```

STRIPS form:

```
control(fuel-nozzle-valve open)
```

Job Guide entry:

13. [1Ø] (B) Using wrenches, close three external fuel tank manual refueling valves.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC close)))
      (ACTOR  man-B)
      (OBJECT external-tank-manual-refueling-valves)
      (SOURCE nil)
      (DESTIN nil)
      (ORGAN  wrenches))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC close)))
      (ACTOR  man-B)
      (OBJECT external-tank-manual-refueling-valves)
      (SOURCE nil)
      (DESTIN close)
      (ORGAN  (AND hand wrenches)))
```

STRIPS form:

```
control(external-tank-manual-refueling-valves close)
```

Job Guide entry:

14. (C) Start fuel pump to begin servicing of fuel.

Conceptual Dependency form:

```
(ACHIEVE-PHYSICAL-STATE (VERB    ((SPECIFIC start)
                                   (STATE operating)))
                        (ACTOR  man-C)
                        (OBJECT fuel-pump)
                        (DESTIN begin-servicing-fuel))
```

CD Operationalized:

```
(ACHIEVE-PHYSICAL-STATE (VERB    ((SPECIFIC start)
                                   (STATE operating)))
                        (ACTOR  man-C)
                        (OBJECT fuel-source
                                ((SPECIFIC fuel-pump)))
                        (DESTIN on)
                        (ORGAN  hand))
```

STRIPS form:

```
control(fuel-source on)
```

Job Guide entry:

15. (B) When fuel flow stops, close nozzle valve.

Conceptual Dependency form:

```
(ATTEND (VERB   ((SPECIFIC implied)
                  (IMPLIED-DEFAULT nil)))
        (ACTOR  man-B)
        (OBJECT fuel-flow-indicator)
        (ORGAN  nil)
        (NOTE   stop-increasing))
(MOVE (VERB   ((SPECIFIC close)))
      (ACTOR  man-B)
      (OBJECT nozzle-valve)
      (SOURCE nil)
      (DESTIN nil))
```

CD Operationalized:

```
(ATTEND (VERB   ((SPECIFIC implied)
                  (IMPLIED-DEFAULT nil)))
        (ACTOR  man-B)
        (OBJECT fuel-flow-indicator)
        (ORGAN  sight)
        (NOTE   stop-increasing))
(MOVE (VERB   ((SPECIFIC close)))
      (ACTOR  man-B)
      (OBJECT fuel-nozzle-valve  ((SPECIFIC nozzle-valve)))
      (SOURCE nil)
      (DESTIN close)
      (ORGAN  hand))
```

STRIPS form:

```
attend(fuel-flow-indicator stop-increasing)
control(fuel-nozzle-valve close)
```

Job Guide entry:

16. (C) Stop fuel pump.

Conceptual Dependency form:

```
(ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC stop)
                                  (STATE not-operating)))
                        (ACTOR  man-C)
                        (OBJECT fuel-pump)
                        (DESTIN nil))
```

CD Operationalized:

```
(ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC stop)
                                  (STATE not-operating)))
                        (ACTOR  man-C)
                        (OBJECT fuel-source
                                ((SPECIFIC fuel-pump)))
                        (DESTIN off)
                        (ORGAN  hand))
```

STRIPS form:

```
control(fuel-source off)
```

Job Guide entry:

17. [1Ø] (B) Using wrenches, open three external fuel tank manual refueling valves.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC open)))
      (ACTOR  man-B)
      (OBJECT external-fuel-tank-manual-refueling-valves)
      (SOURCE nil)
      (DESTIN nil))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC open)))
      (ACTOR  man-B)
      (OBJECT external-fuel-tank-manual-refueling-valves)
      (SOURCE nil)
      (DESTIN open)
      (ORGAN  hand))
```

STRIPS form:

```
control(external-fuel-tank-manual-refueling-valves open)
```

Job Guide entry:

18. (B) Disconnect nozzle.

Conceptual Dependency form:

```
(MOVE (VERB    ((SPECIFIC disconnect)))
      (ACTOR  man-B)
      (OBJECT fuel-nozzle)
      (SOURCE nil)
      (DESTIN nil))
```

CD Operationalized:

```
(MOVE (VERB    ((SPECIFIC disconnect)))
      (ACTOR  man-B)
      (OBJECT fuel-nozzle)
      (SOURCE SPR(F-16))
      (DESTIN floor)
      (ORGAN  hand))
```

STRIPS form:

```
fuel-nozzle-disconnected(SPR(F-16))
```

Job Guide entry:

19. (B) Install cap on refuel adapter.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC install)))
      (ACTOR  man-B)
      (OBJECT fuel-cap)
      (SOURCE nil)
      (DESTIN refuel-adapter))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC install)))
      (ACTOR  man-B)
      (OBJECT fuel-cap(F-16))
      (SOURCE nil)
      (DESTIN SPR(F-16) ((SPECIFIC refuel-adapter)))
      (ORGAN  hand))
```

STRIPS form:

```
twist(fuel-cap(F-16) SPR(F-16))
```

Job Guide entry:

20. (B) Reel up and stow fuel servicing hose.

Conceptual Dependency form:

```
(ACHIEVE-
PHYSICAL-STATE (VERB   ((SPECIFIC reel-up-and-stow)
                         (STATE wrapped-around-reel)))
               (ACTOR  man-B)
               (OBJECT fuel-servicing-hose)
               (SOURCE nil)
               (DESTIN nil))
```

CD Operationalized:

```
(ACHIEVE-
PHYSICAL-STATE (VERB   ((SPECIFIC reel-up-and-stow)
                         (STATE wrapped-around-reel)))
               (ACTOR  man-B)
               (OBJECT fuel-hose
                       ((SPECIFIC fuel-servicing-hose)))
               (SOURCE nil)
               (DESTIN fuel-truck-reel)
               (ORGAN  hand))
```

STRIPS form:

```
reel(fuel-hose)
```

Job Guide entry:

21. (B) Remove static bond cable.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC remove)))
      (ACTOR  man-B)
      (OBJECT static-bond-cable)
      (SOURCE nil)
      (DESTIN nil))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC remove)))
      (ACTOR  man-B)
      (OBJECT static-bond-cable-plug
               ((SPECIFIC static-bond-cable)))
      (SOURCE ground-point(F-16))
      (DESTIN floor)
      (ORGAN  hand))
```

STRIPS form:

```
extract(static-bond-cable-plug ground-point(F-16))
```

Job Guide entry:

22. (B) Remove truck ground cable.

Conceptual Dependency form:

```
(MOVE (VERB   ((SPECIFIC remove)))
      (ACTOR  man-B)
      (OBJECT truck-ground-cable)
      (SOURCE aircraft-earth-ground-point)
      (DESTIN nil))
```

CD Operationalized:

```
(MOVE (VERB   ((SPECIFIC remove)))
      (ACTOR  man-B)
      (OBJECT fuel-source-ground-cable-clamp
              ((SPECIFIC truck-ground-cable)))
      (SOURCE aircraft-earth-ground-point)
      (DESTIN clamp-stowage-point)
      (ORGAN  hand))
```

STRIPS form:

```
attach(fuel-source-ground-cable-clamp clamp-stowage-point)
```

Job Guide entry:

Close and latch access door 31Ø3.

Conceptual Dependency form:

```
(ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC close-and-latch)
                                  (STATE flight-worthy)))
                        (ACTOR  nil)
                        (OBJECT access-door-31Ø3)
                        (DESTIN nil))
```

Conceptual Dependency form:

```
(ACHIEVE-PHYSICAL-STATE (VERB   ((SPECIFIC close-and-latch)
                                  (STATE flight-worthy)))
                        (ACTOR  nil)
                        (OBJECT (door 31Ø3))
                        (DESTIN flight-position)
                        (ORGAN  hand))
```

Appendix D

PostSTRIPS Operators for the Refueling Task

This appendix lists a proposed set of PostSTRIPS Operators that the robot should be able to perform in order to refuel an F-15 and F-16 aircraft without any assistance.

PostSTRIPS is similar to the well publicized STRIPS with the enhancement that "postconditions" are included. Postconditions are explained in Chapter IV. These should not be confused with items in the "add list" which are results of the operator. Consulting a few examples should clear up any confusion.

A nearer term goal of reducing the number of men required to refuel by having a robot assist them would not require the robot to be able to preform all of these tasks. These operators are tightly coupled; any change to one may affect the operation of another. The ordering of conditions may contain temporal information, depending on the operator.

An exclaimation point indicates that this item must be true in the current world and PostSTRIPS is not to try to make it true. A plus sign is used to separate effects from side effects in the add and delete lists.

ATTACH

```
attach(m n):
Robot is to attach m to n.

 Preconditions:
      !ATTACHABLE(m n) &
      GRASPED(m) &
      NEXTTO(m n)
 Postconditions:
      HANDEMPTY
 Delete list: ATTACHED(m $)
              +
              SUPPORTED(m $)
    Add list: ATTACHED(m n)
              +
              SUPPORTED(m n)
```

ATTEND

```
attend(p q):
Robot is to monitor p for condition q.  STRIPS will assume
condition occurs for planning purposes.

 Preconditions:
      NEXTTO(ROBOT p) | !PERVASIVE(p)
 Delete list: STATE(p $)
    Add list: STATE(p q)
```

CLOSE-ACCESS-DOOR (see OPEN-ACCESS-DOOR)

```
close-access-door(m):
Robot is to secure access door m in its flight position.

 Preconditions:
      !ACCESSIBLE(ROBOT m) &
      !OPENABLE(ROBOT m) &
      NEXTTO(ROBOT m)
 Delete list: UNSECURED(m)
    Add list: SECURED(m)
```

CONTROL

```
control(m n):
Change the status of unit m to n.

 Preconditions:
      !POSSIBLE(m n) &
      !ACCESSIBLE(ROBOT m) &
      !CONTROLABLE(ROBOT m) &
      NEXTTO(ROBOT m)
 Delete list: STATUS(m $)
    Add list: STATUS(m n)
```

```
EXTRACT (see INSERT)

extract(m f):
Remove male unit from female recepticle to destination.

 Preconditions:
      !ACCESSIBLE(ROBOT m) &
      !INSERTED(m f) &
      GRASPED(m) &
      (!NOT-TWISTABLE(m) | UNSCREWED(m))
 Delete list: INSERTED(m f) + SUPPORTED(m f)
    Add list: EXTRACTED(m f) + LIFTED(ROBOT m)


FUEL-NOZZLE-CONNECTED (see FUEL-NOZZLE-DISCONNECTED)

fuel-nozzle-connected(m):
Connect fuel nozzle location m.  This adds a precondition of
having the hose unreeled.

 Preconditions:
      STATUS(FUEL-NOZZLE-VALVE CLOSE) &
      EXTRACTED(FUEL-CAP m) &
      SUPPORTED(FUEL-CAP x) &
      UNREELED(FUEL-HOSE) &
      SCREWEDON(FUEL-NOZZLE m)
 Delete list: DISCONNECTED(FUEL-NOZZLE m)
    Add list: CONNECTED(FUEL-NOZZLE m)


FUEL-NOZZLE-DISCONNECTED (see FUEL-NOZZLE-CONNECTED)

fuel-nozzle-disconnected(m):
Disconnect fuel nozzle from location m.

 Preconditions:
      STATUS(FUEL-NOZZLE-VALVE CLOSE) &
      EXTRACTED(FUEL-NOZZLE m) &
      SUPPORTED(FUEL-NOZZLE FLOOR)
 Delete list: CONNECTED(FUEL-NOZZLE m)
    Add list: DISCONNECTED(FUEL-NOZZLE m)
```

NOTE: Connect requires unreeling while disconnecting does not require reeling. This was a somewhat arbitrary choice.

GOTO

goto-place(m):
Robot transports self to location m within a room.

```
 Preconditions:
      !HANDEMPTY & LOCINROOM(m x) & INROOM(ROBOT x)
 Delete list: ATROBOT($) NEXTTO(ROBOT $)
    Add list: ATROBOT(m)
```

goto-item(m):
Robot transports self next to item n. This can be used to get next to an item in a room or to get next to the door to get out of the room.

```
 Precondition:
      !HANDEMPTY &
      [ [INROOM(m x) & INROOM(ROBOT x)] |
        [CONNECTS(d x y) & INROOM(ROBOT x)] ]
 Delete list: ATROBOT($) NEXTTO(ROBOT $)
    Add list: NEXTTO(ROBOT m)
```

GO THROUGH DOOR

go-thru-door1(k l m):
Robot travels from one room to another empty handed.

```
 Preconditions:
      !HANDEMPTY &
      !CONNECTS(k l m) & INROOM(ROBOT l) & NEXTTO(ROBOT k)
 Delete list: INROOM(ROBOT $) + ATROBOT($) NEXTTO(ROBOT $)
    Add list: INROOM(ROBOT m) + NEXTTO(ROBOT k)
```

go-thru-door2(k l m):
Robot travels from one room to another carrying x.

```
 Preconditions:
      !GRASPED(x) &
      !CONNECTS(k l m) & INROOM(ROBOT l) & NEXTTO(ROBOT k)
 Delete list: INROOM(ROBOT $) INROOM(x $) +
              ATROBOT($) NEXTTO(ROBOT $)
    Add list: INROOM(x m) INROOM(ROBOT m) +
              NEXTTO(ROBOT k) NEXTTO(ROBOT x)
```

GRASP (see RELEASE)

```
grasp(m):
Robot is to gain control of item m.

 Preconditions:
      !GRASPABLE(ROBOT m) &
      !ACCESSIBLE(ROBOT m) &
      HANDEMPTY &
      NEXTTO(ROBOT m)
 Delete list: HANDEMPTY
    Add list: GRASPED(m)
```

GROUND

```
ground-aircraft(m):
Electrically connect aircraft m to earth.

 Preconditions:
      GRASPED(ground-wire) & NEXTTO(ROBOT m) &
      INSERTED(PLUG-END-GROUND-WIRE m)
      ATTACHED(CLAMP-END-GROUND-WIRE EARTH-GROUND-POINT)
 Delete list: UNGROUNDED(m)
    Add list: GROUNDED(m)
              AIRCRAFT-EARTH-GROUND-POINT(m))
```

NOTE: GROUND-AIRCRAFT is not part of the REFUELING script but is part of the ENGINE-SHUTDOWN script. This step is part of the refueling script only as a back-up. UNGROUND-AIRCRAFT is part of the BEFORE-STARTING-ENGINES script and is not shown here. AIRCRAFT-EARTH-GROUND-POINT(aircraft) saves the location for this aircraft for the operator ground-fuel-source.

```
ground-fuel-source(m n)
Electrically connect fuel source (pump, truck, etc) to m and
to earth (at same point aircraft is connected to earth, i.e.
n).

 Preconditions:
      ATTACHED(FUEL-SOURCE-GROUND-CABLE-CLAMP n)
      INSERTED(STATIC-BOND-CABLE-PLUG m)
 Delete list: UNGROUNDED-FUEL-SOURCE
    Add list: GROUNDED-FUEL-SOURCE
```

INSERT (see EXTRACT)

insert(m f):
Robot is to insert male into female.

```
 Preconditions:
      !ACCESSIBLE(ROBOT m) & !ACCESSIBLE(ROBOT f) &
      !INSERTABLE(m f) & NEXTTO(m f) & NEXTTO(ROBOT f)
 Postconditions:
      !TWISTABLE(ROBOT m) |
      [!NOT-TWISTABLE(ROBOT m) & HANDEMPTY]
 Delete list: INSERTED(m $) EXTRACTED(m f)
    Add list: INSERTED(m f) + SUPPORTED(m f)
```

LIFT

lift(m):
Pick up the thing that you are grasping.

```
 Preconditions:
     !MOVEABLE(ROBOT m)
 Delete list: SUPPORTED(m $)
    Add list: LIFTED(ROBOT m)
```

MOVE

move(m o):
Use hand to move m to o.

```
 Preconditions:
      !ACCESSIBLE(ROBOT m) & !ACCEPTABLE(m o) &
      !ACCESSIBLE(ROBOT o) & GRASPED(m) &
      [NEXTTO(ROBOT o) | !PERVASIVE(o)]
 Postconditions:
      HANDEMPTY
 Delete list: LOCAL-AT(m $) ATTACHED(m $)
              SUPPORTED(m $) LIFTED(ROBOT m)
    Add list: LOCAL-AT(m o) SUPPORTED(m o)
```

OPEN-ACCESS-DOOR (see CLOSE-ACCESS-DOOR)

open-access-door(m):
Robot is to open an access door on the aircraft.

```
 Preconditions:
      !ACCESSIBLE(ROBOT m) & !OPENABLE(ROBOT m) &
      NEXTTO(ROBOT m)
 Delete list: SECURED(m)
    Add list: UNSECURED(m)
```

```
PTRANS

ptrans-place(m n):
Robot changes location of item m from n.

 Preconditions:
      !ACCESSIBLE(ROBOT m) &
      [!POSITIONABLE(ROBOT m) | MOVEABLE(ROBOT m)] &
      GRASPED(m) &
      [!ROLLS(m) | LIFTED(ROBOT m)] &
      !LOCINROOM(n x) &
      INROOM(ROBOT x)
 Postconditions:
      HANDEMPTY
 Delete list: AT(m $) NEXTTO(m $) NEXTTO($ m) +
              NEXTTO(ROBOT $) ATROBOT($)
    Add list: AT(m n) + ATROBOT(n) NEXTTO(ROBOT m)

ptrans-iteml(m n):
Robot moves item m next to n.

 Preconditions:
      !ACCESSIBLE(ROBOT m) &
      [!POSITIONABLE(ROBOT m) | MOVEABLE(ROBOT m)] &
      GRASPED(m) &
      [!ROLLS(m) | LIFTED(ROBOT m)] &
      [ [INROOM(n x) & INROOM(ROBOT x)] |
        [!CONNECTS(n x y) & INROOM(ROBOT x)]]
 Delete list: AT(m $) NEXTTO(m $) NEXTTO($ m) +
              NEXTTO(ROBOT $) ATROBOT($)
    Add list: NEXTTO(m n) NEXTTO(n m) +
              NEXTTO(ROBOT m) NEXTTO(ROBOT n)


ptrans-item2(m n):
Robot moves item m next to n.
 Preconditions:
      !GRASPED(m) &
      !ACCESSIBLE(ROBOT m) &
      [!POSITIONABLE(ROBOT m) | MOVEABLE(ROBOT m)] &
      [!ROLLS(m) | LIFTED(ROBOT m)] &
      [ [INROOM(n x) & INROOM(ROBOT x)] |
        [!CONNECTS(n x y) & INROOM(ROBOT x)]]
 Delete list: AT(m $) NEXTTO(m $) NEXTTO($ m) +
              NEXTTO(ROBOT $) ATROBOT($)
    Add list: NEXTTO(ROBOT n) + NEXTTO(m n)
              NEXTTO(n m) NEXTTO(ROBOT m)
```

```
REEL (see UNREEL)

reel(m):
Robot is to stow item m on its reel.

 Preconditions:
      !REELABLE(ROBOT m) &
      !ACCESSIBLE(ROBOT m) &
      NEXTTO(ROBOT m)
 Delete list: UNREELED(m)
    Add list: REELED(m)
              +
              NEXTTO(ROBOT REEL)


RELEASE (see GRASP)

release(m):
Robot is to relinquish control of item m.

 Preconditions:
      !GRASPED(m) &
      !ACCEPTABLE(m x) &
      !SUPPORTED(m x)
 Delete list: GRASPED(m)
              +
              LIFTED($)
    Add list: HANDEMPTY


TWIST (see UNTWIST)

twist(m n):
Robot is to turn m clockwise about n.

 Preconditions:
      !TWISTABLE(ROBOT m) &
      !ACCESSIBLE(ROBOT m) &
      !ACCESSIBLE(ROBOT n) &
      INSERTED(m n)
 Postconditions:
      HANDEMPTY
 Delete list: UNSCREWED(m n)
              +
              SUPPORTED(m $)
    Add list: SCREWEDON(m n)
              +
              SUPPORTED(m n)
```

UNREEL (see REEL)

```
unreel(m):
The robot is to remove item m from its reel at n.

 Preconditions:
      !REELABLE(ROBOT m) &
      !ACCESSIBLE(ROBOT m) &
      NEXTTO(ROBOT m)
 Delete list: REELED(m)
    Add list: UNREELED(m)
              +
              NEXTTO(ROBOT reel)
```

UNTWIST (see TWIST)

```
untwist(m n):
Robot is to turn m counterclockwise about n.

 Preconditions:
      !TWISTABLE(ROBOT m) &
      !ACCESSIBLE(ROBOT m) &
      !ACCESSIBLE(ROBOT n) &
      NEXTTO(ROBOT m) &
      GRASPED(m)
 Delete list: SCREWEDON(m n)
    Add list: UNSCREWED(m n)
```

Appendix E

Initial PostSTRIPS World Model

This appendix shows the world model prior to refueling.

```
-- WORLD STATIC --
ACCEPTABLE(FIRE-EXTINGUISHER FLOOR)
ACCEPTABLE(FUEL-CAP($) FLOOR)
ACCEPTABLE(FUEL-CAP(x) SPR(x))
ACCEPTABLE(FUEL-CAP($) STOWAGE-POINT)
ACCEPTABLE(FUEL-NOZZLE FLOOR)
ACCEPTABLE(FUEL-NOZZLE SPR)
ACCEPTABLE(FUEL-SOURCE-GROUND-CABLE-CLAMP
           AIRCRAFT-EARTH-GROUND-POINT($))
ACCEPTABLE(GROUND-WIRE STORAGE-BIN)
ACCEPTABLE(GROUND-WIRE FLOOR)
ACCEPTABLE(STATIC-BOND-CABLE-PLUG GROUND-POINT($))
ATTACHABLE(FUEL-SOURCE-GROUND-CABLE-CLAMP
           AIRCRAFT-EARTH-GROUND-POINT($))
CONNECTS(DOOR1 MAINTENANCE-SHACK FLIGHTLINE)
CONNECTS(DOOR1 FLIGHTLINE MAINTENANCE-SHACK)
INSERTABLE(FUEL-CAP(x) SPR(x))
INSERTABLE(FUEL-NOZZLE SPR($))
INSERTABLE(STATIC-BOND-CABLE-PLUG GROUND-POINT($))
LOCINROOM(between(AIRCRAFT($) FUEL-SOURCE) FLIGHTLINE)
LOCINROOM(CLAMP-STOWAGE-POINT FLIGHTLINE)
LOCINROOM(STORAGE-BIN MAINTENANCE-SHACK)
LOCINROOM(upwind(AIRCRAFT($)) FLIGHTLINE)
PERVASIVE(AREA)
PERVASIVE(FLOOR)
PERVASIVE(STOWAGE-POINT)
POSSIBLE(EXTERNAL-TANK-REFUEL-VALVES OPEN)
POSSIBLE(EXTERNAL-TANK-REFUEL-VALVES CLOSE)
POSSIBLE(FUEL-NOZZLE-VALVE OPEN)
POSSIBLE(FUEL-NOZZLE-VALVE CLOSE)
POSSIBLE(FUEL-SOURCE ON)
POSSIBLE(FUEL-SOURCE OFF)
ROLLS(FIRE-EXTINGUISHER)
SUPPORTED(FIRE-EXTINGUISHER FLOOR)

-- WORLD DYNAMIC --
AT(FUEL-SOURCE upwind(AIRCRAFT(F-16)))
AT(GROUND-WIRE STORAGE-BIN)
INROOM(AIRCRAFT-EARTH-GROUND-POINT($) FLIGHTLINE)
INROOM(FIRE-EXTINGUISHER FLIGHTLINE)
INROOM(FUEL-FLOW-INDICATOR FLIGHTLINE)
INROOM(FUEL-HOSE FLIGHTLINE)
INROOM(FUEL-NOZZLE FLIGHTLINE)
INROOM(FUEL-NOZZLE-VALVE FLIGHTLINE)
INROOM(FUEL-TYPE FLIGHTLINE)
INROOM(FUEL-SOURCE FLIGHTLINE)
INROOM(FUEL-SOURCE-GROUND-CABLE-CLAMP FLIGHTLINE)
INROOM(GROUND-WIRE MAINTENANCE-SHACK)
INROOM(QUICK-DISCONNECT FLIGHTLINE)
INROOM(STATIC-BOND-CABLE-PLUG FLIGHTLINE)
REELED(FUEL-HOSE)
STATUS(FUEL-NOZZLE-VALVE CLOSE)
STATUS(FUEL-SOURCE OFF)

-- ROBOT STATIC --
ACCESSIBLE(ROBOT AIRCRAFT-EARTH-GROUND-POINT($))
ACCESSIBLE(ROBOT CLAMP-STOWAGE-POINT)
ACCESSIBLE(ROBOT DOOR(29))
ACCESSIBLE(ROBOT DOOR(3103))
ACCESSIBLE(ROBOT FIRE-EXTINGUISHER)
ACCESSIBLE(ROBOT FLOOR)
ACCESSIBLE(ROBOT FUEL-CAP(F-15))
ACCESSIBLE(ROBOT FUEL-CAP(F-16))
ACCESSIBLE(ROBOT FUEL-FLOW-INDICATOR))
ACCESSIBLE(ROBOT FUEL-HOSE)
ACCESSIBLE(ROBOT FUEL-NOZZLE)
ACCESSIBLE(ROBOT FUEL-NOZZLE-VALVE)
ACCESSIBLE(ROBOT FUEL-TYPE)
ACCESSIBLE(ROBOT FUEL-SOURCE)
ACCESSIBLE(ROBOT FUEL-SOURCE-GROUND-CABLE-CLAMP)
ACCESSIBLE(ROBOT FUEL-VENT(F-15))
ACCESSIBLE(ROBOT GROUND-POINT($))
ACCESSIBLE(ROBOT GROUND-WIRE)
ACCESSIBLE(ROBOT SPR(F-15))
ACCESSIBLE(ROBOT SPR(F-16))
ACCESSIBLE(ROBOT STATIC-BOND-CABLE-PLUG)
ACCESSIBLE(ROBOT STORAGE-BIN)
ACCESSIBLE(ROBOT STOWAGE-POINT)
CONTROLABLE(ROBOT FUEL-NOZZLE-VALVE)
CONTROLABLE(ROBOT FUEL-SOURCE)
GRASPABLE(ROBOT FIRE-EXTINGUISHER)
GRASPABLE(ROBOT FUEL-CAP($))
GRASPABLE(ROBOT FUEL-NOZZLE)
GRASPABLE(ROBOT FUEL-SOURCE-GROUND-CABLE-CLAMP)
GRASPABLE(ROBOT GROUND-WIRE)
GRASPABLE(ROBOT GROUND-WIRE)
GRASPABLE(ROBOT STATIC-BOND-CABLE-PLUG)
MOVEABLE(ROBOT FUEL-CAP($))
MOVEABLE(ROBOT FUEL-NOZZLE)
MOVEABLE(ROBOT FUEL-SOURCE-GROUND-CABLE-CLAMP)
MOVEABLE(ROBOT STATIC-BOND-CABLE-PLUG)
NOT-TWISTABLE(ROBOT STATIC-BOND-CABLE-PLUG)
OPENABLE(ROBOT DOOR(29))
OPENABLE(ROBOT DOOR(3103))
POSITIONABLE(ROBOT FUEL-SOURCE)
POSITIONABLE(ROBOT FIRE-EXTINGUISHER)
REELABLE(ROBOT FUEL-HOSE)
```

```
TWISTABLE(ROBOT FUEL-CAP($))
TWISTABLE(ROBOT FUEL-NOZZLE)

-- ROBOT DYNAMIC --
ATROBOT(CORNER)
INROOM(ROBOT MAINTENANCE-SHACK)
HANDEMPTY

-- F-15 --
AIRCRAFT-EARTH-GROUND-POINT(G7)
GROUNDED(F-15)
INROOM(DOOR(29) FLIGHTLINE)
INROOM(FUEL-CAP(F-15) FLIGHTLINE)
INROOM(FUEL-VENT(F-15) FLIGHTLINE)
INROOM(GROUND-POINT(F-15) FLIGHTLINE)
INROOM(SPR(F-15) FLIGHTLINE)
INSERTED(FUEL-CAP(F-15) SPR(F-15))
NEXTTO(AIRCRAFT(F-15) FIRE-EXTINGUISHER)
NEXTTO(AIRCRAFT(F-15) FUEL-SOURCE)
SCREWEDON(FUEL-CAP(F-15) SPR(F-15))
SECURED(DOOR(29))

-- F-16 --
AIRCRAFT-EARTH-GROUND-POINT(G4)
GROUNDED(F-16)
INROOM(DOOR(31Ø3) FLIGHTLINE)
INROOM(EXTERNAL-TANK-MANUAL-REFUEL-VALVES(F-16) FLIGHTLINE)
INROOM(FUEL-CAP(F-16) FLIGHTLINE)
INROOM(FUEL-VENT(F-16) FLIGHTLINE)
INROOM(GROUND-POINT(F-16) FLIGHTLINE)
INROOM(SPR(F-16) FLIGHTLINE)
INSERTED(FUEL-CAP(F-16) SPR(F-16))
NEXTTO(AIRCRAFT(F-16) FIRE-EXTINGUISHER)
NEXTTO(AIRCRAFT(F-16) FUEL-SOURCE)
SCREWEDON(FUEL-CAP(F-16) SPR(F-16))
UNSECURED(DOOR(31Ø3))
```

Appendix F

Unplanned and Planned Refueling Scripts

This appendix shows the F-15 refueling script, the F-16 refueling script, the F-15 script generated by a planner, and the F-16 script generated by a planner.

F-15 Refueling script before planning. Some of these have been slightly modified to improve reabability. They are marked with a "*".

```
((case-IAC-EECORP (AIRCRAFT F-15))
 (ptrans-place FIRE-EXTINGUISHER
               (between (AIRCRAFT F-15) FUEL-SOURCE))
 (ground-aircraft (AIRCRAFT F-15))
 (ground-fuel-source FUEL-SOURCE
                     (AIRCRAFT-EARTH-GROUND-POINT F-15))
 (open-access-door (DOOR 29))
 (extract (FUEL-CAP F-15) (SPR F-15))
 (move (FUEL-CAP F-15) STOWAGE-POINT)
 (fuel-nozzle-connected (SPR F-15))
 (control FUEL-NOZZLE-VALVE OPEN)
 (attend FUEL-NOZZLE ENGAGEMENT)
 (attend QUICK-DISCONNECT ENGAGEMENT)
 (goto-item FIRE-EXTINGUISHER)
 (attend FUEL-TYPE JP-4)*
 (control FUEL-SOURCE ON)
 (attend (FUEL-VENT F-15) AIRFLOW)*
 (case-airflow-difficulty (FUEL-VENT-MASTS F-15))
 (case-fuel-spill (AIRCRAFT F-15))
 (case-fuel-spill-repair (AIRCRAFT F-15))
 (case-fuel-system-calibration-check (AIRCRAFT F-15))
 (attend FUEL-FLOW-INDICATOR STOP-INCREASING)
 (control FUEL-SOURCE OFF)
 (fuel-nozzle-disconnected (SPR F-15))
 (twist (FUEL-CAP F-15) (SPR F-15))
 (attend AREA FOREIGN-OBJECTS)
 (close-access-door (DOOR 29)))
```

F-16 Refueling script before planning.

```
((ground-aircraft (GROUND-POINT F-16))
 (ptrans-place FUEL-SOURCE (upwind (AIRCRAFT F-16)))
 (attach FUEL-SOURCE-GROUND-CABLE-CLAMP
         (AIRCRAFT-EARTH-GROUND-POINT F-16))
 (insert STATIC-BOND-CABLE-PLUG (GROUND-POINT F-16))
 (ptrans-place FIRE-EXTINGUISHER
         (between (AIRCRAFT F-16) FUEL-SOURCE))
 (attend (FUEL-VENT F-16) OBSTRUCTION-FREE)
 (attend FUEL-NOZZLE (and (not DAMAGE) SERVICEABLE))
 (unreel FUEL-HOSE)
 (extract (FUEL-CAP F-16) (SPR F-16))
 (move (FUEL-CAP F-16) FLOOR)
 (attend (SPR F-16) (and (not DAMAGE) SERVICEABLE))
 (fuel-nozzle-connected (SPR F-16))
 (control FUEL-NOZZLE-VALVE OPEN)
 (control EXTERNAL-TANK-MANUAL-REFUELING-VALVES CLOSE)
 (control FUEL-SOURCE ON)
 (attend FUEL-FLOW-INDICATOR STOP-INCREASING)
 (control FUEL-NOZZLE-VALVE CLOSE)
 (control FUEL-SOURCE OFF)
 (control (EXTERNAL-TANK-MANUAL-REFUEL-VALVESF-16) OPEN)
 (fuel-nozzle-disconnected (SPR F-16))
 (twist (FUEL-CAP F-16) (SPR F-15))
 (reel FUEL-HOSE)
 (extract STATIC-BOND-CABLE-PLUG (GROUND-POINT F-16))
 (attach FUEL-SOURCE-GROUND-CABLE-CLAMP CLAMP-STOWAGE-POINT)
 (close-access-door (DOOR 31Ø3)))
```

F-15 Refueling script after planning. Note that operators which have no add or delete lists are short scripts and once they are filled out they are dropped from the plan.

```
((case-IAC-EECORP (AIRCRAFT F-15))
 (goto-item DOOR1)
 (go-thru-door DOOR1 MAINTENANCE-SHACK FLIGHTLINE)
 (goto-item FIRE-EXTINGUISHER)
 (grasp FIRE-EXTINGUISHER)
 (ptrans-place FIRE-EXTINGUISHER
              (between (AIRCRAFT F-15) FUEL-SOURCE))
 (release FIRE-EXTINGUISHER)
 (goto-item FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (grasp FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (lift FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (ptrans-iteml FUEL-SOURCE-GROUND-CABLE-CLAMP
              (AIRCRAFT-EARTH-GROUND-POINT F-15))
 (attach FUEL-SOURCE-GROUND-CABLE-CLAMP
        (AIRCRAFT-EARTH-GROUND-POINT F-15))
 (release FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (goto-item STATIC-BOND-CABLE-PLUG)
 (grasp STATIC-BOND-CABLE-PLUG)
 (lift STATIC-BOND-CABLE-PLUG)
 (ptrans-iteml STATIC-BOND-CABLE-PLUG (GROUND-POINT F-16))
 (insert STATIC-BOND-CABLE-PLUG (GROUND-POINT F-16))
 (release STATIC-BOND-CABLE-PLUG)
 (goto-item (DOOR 29))
 (open-access-door (DOOR 29))
 (goto-item (FUEL-CAP F-15))
 (grasp (FUEL-CAP F-15))
 (untwist (FUEL-CAP F-15) (SPR F-15))
 (extract (FUEL-CAP F-15) (SPR F-15))
 (move (FUEL-CAP F-15) STOWAGE-POINT)
 (release (FUEL-CAP F-15))
 (goto-item FUEL-HOSE)
 (unreel FUEL-HOSE)
 (goto-item FUEL-NOZZLE)
 (grasp FUEL-NOZZLE)
 (lift FUEL-NOZZLE)
 (ptrans-iteml FUEL-NOZZLE (SPR F-15))
 (insert FUEL-NOZZLE (SPR F-15))
 (twist FUEL-NOZZLE (SPR F-15))
 (release FUEL-NOZZLE)
 (goto-item FUEL-NOZZLE-VALVE)
 (control FUEL-NOZZLE-VALVE OPEN)
 (goto-item FUEL-NOZZLE)
 (attend FUEL-NOZZLE ENGAGEMENT)
 (goto-item QUICK-DISCONNECT)
 (attend QUICK-DISCONNECT ENGAGEMENT)
 (goto-item FIRE-EXTINGUISHER)
 (goto-item FUEL-TYPE)
 (attend FUEL-TYPE JP-4)
 (goto-item FUEL-SOURCE)
(control FUEL-SOURCE ON)
(goto-item (FUEL-VENT F-15))
(attend (FUEL-VENT F-15) AIRFLOW)
(case-airflow-difficulty (FUEL-VENT-MASTS F-15))
(case-fuel-spill (AIRCRAFT F-15))
(case-fuel-spill-repair (AIRCRAFT F-15))
(case-fuel-system-calibration-check (AIRCRAFT F-15))
(goto-item FUEL-FLOW-INDICATOR)
(attend FUEL-FLOW-INDICATOR STOP-INCREASING)
(goto-item FUEL-SOURCE)
(control FUEL-SOURCE OFF)
(goto-item FUEL-NOZZLE-VALVE)
(control FUEL-NOZZLE-VALVE CLOSE)
(goto-item FUEL-NOZZLE)
(grasp FUEL-NOZZLE)
(untwist FUEL-NOZZLE)
(extract FUEL-NOZZLE (SPR F-15))
(move FUEL-NOZZLE FLOOR)
(release FUEL-NOZZLE)
(goto-item (FUEL-CAP F-15))
(grasp (FUEL-CAP F-15))
(lift (FUEL-CAP F-15))
(goto-item (SPR F-15))
(insert (FUEL-CAP F-15) (SPR F-15))
(twist (FUEL-CAP F-15) (SPR F-15))
(release (FUEL-CAP F-15))
(attend AREA FOREIGN-OBJECTS)
(goto-item (DOOR 29))
(close-access-door (DOOR 29)))
```

F-16 Refueling script after planning. See F-15 script note.

```
((goto-item DOOR1)
 (go-thru-door DOOR1 MAINTENANCE-SHACK FLIGHTLINE)
 (goto-item FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (grasp FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (lift FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (ptrans-item FUEL-SOURCE-GROUND-CABLE-CLAMP
              (AIRCRAFT-EARTH-GROUND-POINT F-16))
 (attach FUEL-SOURCE-GROUND-CABLE-CLAMP
         (AIRCRAFT-EARTH-GROUND-POINT F-16))
 (release FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (goto-item STATIC-BOND-CABLE-PLUG)
 (grasp STATIC-BOND-CABLE-PLUG)
 (lift STATIC-BOND-CABLE-PLUG)
 (ptrans-item1 STATIC-BOND-CABLE-PLUG (GROUND-POINT F-16))
 (insert STATIC-BOND-CABLE-PLUG (GROUND-POINT F-16))
 (release STATIC-BOND-CABLE-PLUG)
 (goto-item FIRE-EXTINGUISHER)
 (grasp FIRE-EXTINGUISHER)
 (ptrans-place FIRE-EXTINGUISHER
               (between (AIRCRAFT F-16) FUEL-SOURCE))
 (release FIRE-EXTINGUISHER)
 (goto-item (FUEL-VENT F-16))
 (attend (FUEL-VENT F-16) OBSTRUCTION-FREE)
 (goto-item FUEL-NOZZLE)
 (attend FUEL-NOZZLE (and (not DAMAGE) SERVICABLE))
 (goto-item FUEL-HOSE)
 (unreel FUEL-HOSE)
 (goto-item (FUEL-CAP F-16))
 (grasp (FUEL-CAP F-16))
 (untwist (FUEL-CAP F-16) (SPR F-16))
 (extract (FUEL-CAP F-16) (SPR F-16))
 (move (FUEL-CAP F-16) STOWAGE-POINT)
 (release (FUEL-CAP F-16))
 (goto-item (SPR F-16))
 (attend (SPR F-16) (and (not DAMAGE) SERVICABLE))
 (goto-item FUEL-NOZZLE)
 (grasp FUEL-NOZZLE)
 (lift FUEL-NOZZLE)
 (ptrans-item1 FUEL-NOZZLE (SPR F-16))
 (insert FUEL-NOZZLE (SPR F-16))
 (twist FUEL-NOZZLE (SPR F-16))
 (release FUEL-NOZZLE)
 (goto-item FUEL-NOZZLE-VALVE)
 (control FUEL-NOZZLE-VALVE OPEN)
 (goto-item EXTERNAL-TANK-MANUAL-REFUELING-VALVES)
 (control EXTERNAL-TANK-MANUAL-REFUELING-VALVES CLOSE)
 (goto-item FUEL-SOURCE)
 (control FUEL-SOURCE ON)
 (goto-item FUEL-FLOW-INDICATOR)
 (attend FUEL-FLOW-INDICATOR STOP-INCREASING)
 (goto-item FUEL-NOZZLE-VALVE)
(control FUEL-NOZZLE-VALVE CLOSE)
(goto-item FUEL-SOURCE)
(control FUEL-SOURCE OFF)
(goto-item EXTERNAL-TANK-MANUAL-REFUELING-VALVES)
(control EXTERNAL-TANK-MANUAL-REFUELING-VALVES OPEN)
(goto-item FUEL-NOZZLE)
(grasp FUEL-NOZZLE)
(untwist FUEL-NOZZLE (SPR F-16))
(extract FUEL-NOZZLE (SPR F-16))
(move FUEL-NOZZLE FLOOR)
(release FUEL-NOZZLE)
(goto-item (FUEL-CAP F-16))
(grasp (FUEL-CAP F-16))
(lift (FUEL-CAP F-16))
(ptrans-iteml (FUEL-CAP F-16) (SPR F-16))
(insert (FUEL-CAP F-16) (SPR F-16))
(twist (FUEL-CAP F-16) (SPR F-16))
(release (FUEL-CAP F-16))
(goto-item FUEL-HOSE)
(reel FUEL-HOSE)
(goto-item STATIC-BOND-CABLE-PLUG)
(grasp STATIC-BOND-CABLE-PLUG)
(extract STATIC-BOND-CABLE-PLUG (GROUND-POINT F-16))
(move STATIC-BOND-CABLE-PLUG FLOOR)
(release STATIC-BOND-CABLE-PLUG)
(goto-item FUEL-SOURCE-GROUND-CABLE-CLAMP)
(grasp FUEL-SOURCE-GROUND-CABLE-CLAMP)
(lift FUEL-SOURCE-GROUND-CABLE-CLAMP)
(ptrans-iteml STOWAGE-POINT)
(attach FUEL-SOURCE-GROUND-CABLE-CLAMP CLAMP-STOWAGE-POINT)
(release FUEL-SOURCE-GROUND-CABLE-CLAMP)
(goto-item (DOOR 31Ø3))
(close-access-door (DOOR 31Ø3)))
```

Appendix G

Sequential Commonalty of Scripts

The following appendix contains the result of comparing the unplanned F-15 and F-16 refueling scripts and the result of comparing the planned F-15 and F-16 refueling scripts. The first two entries of the first comparison result are a pair indicating two steps that occur in both plans but are transposed from one plan to the other.

Results of comparing scripts prior to planning:

```
((((PTRANS-PLACE FIRE-EXTINGUISHER
               BETWEEN(AIRCRAFT FUEL-SOURCE))
  ((GROUND-AIRCRAFT AIRCRAFT)))
 (EXTRACT FUEL-CAP SPR)
 (MOVE FUEL-CAP STOWAGE-POINT)
 (FUEL-NOZZLE-CONNECTED SPR)
 (CONTROL FUEL-NOZZLE-VALVE OPEN)
 (CONTROL FUEL-SOURCE ON)
 (ATTEND FUEL-FLOW-INDICATOR STOP-INCREASING)
 (CONTROL FUEL-SOURCE OFF)
 (FUEL-NOZZLE-DISCONNECTED SPR)
 (TWIST FUEL-CAP SPR)
 (CLOSE-ACCESS-DOOR DOOR))
```

Results of comparing scripts after planning:

```
((GOTO-ITEM DOOR1)
 (GO-THRU-DOOR DOOR1 MAINTENANCE-SHACK FLIGHTLINE)
 (GOTO-ITEM FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (GRASP FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (LIFT FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (PTRANS-ITEM1 FUEL-SOURCE-GROUND-CABLE-CLAMP
               AIRCRAFT-EARTH-GROUND-POINT)
 (ATTACH FUEL-SOURCE-GROUND-CABLE-CLAMP
         AIRCRAFT-EARTH-GROUND-POINT)
 (RELEASE FUEL-SOURCE-GROUND-CABLE-CLAMP)
 (GOTO-ITEM STATIC-BOND-CABLE-PLUG)
 (GRASP STATIC-BOND-CABLE-PLUG)
 (LIFT STATIC-BOND-CABLE-PLUG)
 (INSERT STATIC-BOND-CABLE-PLUG GROUND-POINT)
 (RELEASE STATIC-BOND-CABLE-PLUG)
 (GOTO-ITEM FUEL-CAP)
 (GRASP FUEL-CAP)
 (UNTWIST FUEL-CAP SPR)
 (EXTRACT FUEL-CAP SPR)
 (MOVE FUEL-CAP STOWAGE-POINT)
 (RELEASE FUEL-CAP)
 (GOTO-ITEM FUEL-NOZZLE)
 (GRASP FUEL-NOZZLE)
 (LIFT FUEL-NOZZLE)
 (PTRANS-ITEM1 FUEL-NOZZLE SPR)
 (INSERT FUEL-NOZZLE SPR)
 (TWIST FUEL-NOZZLE SPR)
 (RELEASE FUEL-NOZZLE)
 (GOTO-ITEM FUEL-NOZZLE-VALVE)
 (CONTROL FUEL-NOZZLE-VALVE OPEN)
 (GOTO-ITEM FUEL-SOURCE)
 (CONTROL FUEL-SOURCE ON)
 (GOTO-ITEM FUEL-FLOW-INDICATOR)
 (ATTEND FUEL-FLOW-INDICATOR STOP-INCREASING)
 (GOTO-ITEM FUEL-SOURCE)
 (CONTROL FUEL-SOURCE OFF)
 (GOTO-ITEM FUEL-NOZZLE)
 (GRASP FUEL-NOZZLE)
 (UNTWIST FUEL-NOZZLE SPR)
 (EXTRACT FUEL-NOZZLE SPR)
 (MOVE FUEL-NOZZLE FLOOR)
 (RELEASE FUEL-NOZZLE)
 (GOTO-ITEM FUEL-CAP)
 (GRASP FUEL-CAP)
 (LIFT FUEL-CAP)
 (INSERT FUEL-CAP SPR)
 (TWIST FUEL-CAP SPR)
 (RELEASE FUEL-CAP)
 (GOTO-ITEM DOOR)
 (CLOSE-ACCESS-DOOR DOOR))
```

Bibliography

1. Asimov, Isaac. I, Robot. New York: Ballantine Books, 1983.

2. Atkinson, Richard C. and others. An Introduction to Mathematical Learning Theory. New York: John Wiley and Sons, Inc., 1965.

3. Barr, Avron, and Edward A. Feigenbaum. Handbook of Artificial Intelligence, Volume 1. Stanford: Heuristech Press, 1981.

4. -----. Handbook of Artificial Intelligence, Volume 2. Stanford: Heuristech Press, 1981.

5. Buchanan, Bruce G. Models of Learning Systems. National Institutes of Health Grant No. 5R24 RR 00612-09. Departments of Computer Science and Electrical Engineering, Stanford University, Stanford CA., January 1981.

6. Carbonell, Jaime G. and others. "An Overview of Machine Learning," Machine Learning: An Artificial Intelligence Approach, edited by Ryszard S. Michalski and others. Palo Alto: Tioga Publishing Co., 1983.

7. Chang, Chin-Liang and Richard Char-Tung Lee Symbolic Logic and Mechanical Theorem Proving. New York: Academic Press, 1973.

8. Clifford, Thomas E. and Hubert G. Schneider III. Creating a Mobile Autonomous Robot Research System (MARRS). MS thesis, AFIT/GE/ENG/84D-19. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1984 (AD-A151 963).

9. Cohen, Paul R., and Edward A. Feigenbaum. Handbook of Artificial Intelligence, Volume 3. Stanford: Heuristech Press, 1981.

10. Davis, Randall. "Expert Systems: Where Are We? And Where Do We Go From Here?," The AI Magazine, 4: 3-22 (Spring 1982).

11. D'Azzo, John J. and Constantine H. Houpis. Linear Control System Analysis and Design (Second Edition). New York: McGraw Hill Book Company, 1981.

12. DeJong, Gerald. "Automatic Schema Acquisition in a Natural Language Environment," The Proceedings of the National Conference on Artificial Intelligence, 18-20 Aug 1982. 410-417. AAAI, Pittsburg, 1982.

13. Department of the Air Force. Technical Handbook - Job Guide - Servicing. T.O. 1F-15A-2-12JG-10-2. Washington: HQ USAF, 4 Nov 1983 (revised 1 Oct 1984).

14. -----. Technical Handbook - Job Guide - Servicing. T.O. 1F-16A-2-12JG-00-1. Washington: HQ USAF, 15 Jul 1981 (revised 15 Jun 85).

15. Derthick, Mark, Masters Candidate, Carnegie-Mellon University. Personal interview. Robitics Institute, Carnegie-Mellon University, Pittsburg, PA, 1 August 1985.

16. Estes, William K. Models of Learning, Memory and Choice. New York: Praeger Publishers, 1982.

17. Fikes, Richard E. and others. "Learning and Executing Generalized Robot Plans," Artificial Intelligence, 3: 251-288 (Winter 1972).

18. Fikes, Richard E. and Nils J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," Artificial Intelligence, 2: 189-208 (Winter 1971).

19. Fuller, John G. "Death by Robot," Omni, 6: 45-46$^+$ (March 1984).

20. Gruver, William A. and others. "Evaluation of Commercially Available Robot Programming Languages," 13th International Symposium on Industrial Robots and Robots 7: 12-58 - 12-68. Society of Manufacturing Engineers, Dearborn, 1983.

21. Johnson, Otto T. Information Please Almanac. Boston: Houghton Mifflin Co., 1985.

22. Kabrisky, Matthew, Professor, Air Force Institute of Technology. Personal interview. School of Engineering, Air Force Institute of Technology, Wright-Patterson AFB OH, 1 November 1985.

23. Korf, Richard E. "Macro-Operators: A Weak Method for Learning," Artificial Intelligence, 26: 35-77 (April 1985).

24. Langley, Patrick W. "Representational Issues in Learning Systems," Computer, 16: 47-51 (October 1983).

25. Langley, Patrick W. and Jaime G. Carbonell. Approaches to Machine Learning. Contract N00014-82-C-50767. The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA, February 1984.

26. Langley, Patrick W. and others. "Rediscovering Chemistry with the Bacon System," Machine Learning: An Artificial Intelligence Approach, edited by Rysard S. Michalski and others. Palo Alto: Tioga Publishing Company, 1983.

27. Levas, A. and M. Selfridge. "Voice Communication with Robots," 13th International Symposium on Industrial Robots and Robots 7. 12-79 - 12-83. Society of Manufacturing Engineers, Dearborn, 1983.

28. Monaghan, Glen E. Navigation For An Autonomous Mobile Robot. MS thesis, AFIT/GE/ENG/84D-47. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1984.

29. Moravec, Hans. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. PhD thesis. Computer Science Department, Stanford University, Stanford CA, September 1980 (AD-A092 604).

30. -----. "The Rovers," Robotics, edited by Marvin Minsky. Garden City, NY: Anchor Press / Doubleday, 1985.

31. Mostow, David Jack. "Machine Transformation of Advice into a Heuristic Search Procedure," Machine Learning: An Artificial Intelligence Approach, edited by Rysard S. Michalski and others. Palo Alto: Tioga Publishing Company, 1983.

32. Norman, M. Frank. Markov Processes and Learning Models. New York: Academic Press, Inc., 1972.

33. Owen, Randall J. III. Environmental Mapping by a Hero-1 Robot Using Sonar and a Laser Barcode Scanner. MS thesis, AFIT/GE/EE/83D-52. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1983 (AD-A138 348).

34. Routh, Richard L. Cortical Thought Theory: A Working Model of the Human Gestalt Mechanism. PhD dissertation. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, in press.

35. Sacerdoti, Earl D. "Planning in a Hierarchy of Abstraction Spaces," Artificial Intelligence, 5: 115-135 (Summer 1974).

36. Schank, Roger C. Reading and Understanding: Teaching from the Perspective of Artificial Intelligence. Hillsdale NJ: Lawrence Erlbaum Associates, Inc, Publishers, 1982.

37. Schank, Roger C. and Christopher K. Riesbeck. Inside Computer Understanding. Hillsdale, NJ: Lawrence Erlbaum Associates, Publishers, 1981.

38. Schank, Roger and Robert Abelson. Scripts Plans Goals and Understanding. Hillsdale NJ: Lawrence Erlbaum Associates, Inc, Publishers, 1977.

39. Taylor, Roslyn J. An Android Research and Development Program. MS thesis, AFIT/GE/EE/83M-3. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, December 1983.

40. Trabasso, Tom and Gordon H. Bower. Attention in Learning: Theory and Research. New York: John Wiely and Sons, Inc., 1968.

41. Winston, Patrick H. Artificial Intelligence. Reading: Addison-Wesley Publishing Co, 1984.

42. -----. "Learning and Reasoning by Analogy," Communications of the ACM, 23: 689-703 (December 1980).

43. -----. "Learning by Creating Transfer Frames," Artificial Intelligence, 10: 147-172 (April 1978).

44. Wittrock, M. C. and others. The Human Brain. Englewood Cliffs NJ: Prentice-Hall, Inc, 1977.

Vita

Captain William Mark Clifford was born 8 March 1955 to Donald D. Clifford and Mary T. Clifford in Geneva, Ill. Raised in Florida, he graduated in 1973 from high school in Avon Park, Florida. In Dec 1977 he received a B. S. in Electrical Engineering from The University of Florida under an Air Force ROTC Scholarship. Upon graduation he received a Reserve commission but remained to work on campus at The Northeast Regional Data Center as a Systems Programmer while waiting to enter active duty.

In May 1978 he entered pilot training at Columbus AFB Mississippi. On graduation he was assigned to fly C-141s with the 76th Military Airlift Squadron at Charleston AFB South Carolina. He upgraded to Instructor Pilot and volunteered to transfer to the 437th Military Airlift Wing as a Simulator Instructor Pilot.

In May 1984, Captain Clifford entered the Air Force Institute of Technology, Wright-Patterson AFB, Ohio as a Masters Candidate in Electrical Engineering. In Oct 1984 he received his Regular commission. In Jan 1985 he was inducted into Tau Beta Pi and Eta Kappa Nu, where he currently serves as president.

Mark married the Linda Terese on 5 May 1984.

Address: 2Ø5 Fairway Drive
Avon Park, Florida 33825

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

AD-A163 829

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/85D-7	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering — 6b. OFFICE SYMBOL *(If applicable)* AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS *(City, State and ZIP Code)* Air Force Institute of Technology Wright-Patterson AFB, OH 45433	7b. ADDRESS *(City, State and ZIP Code)*
8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL *(If applicable)*	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. — PROJECT NO. — TASK NO. — WORK UNIT NO.
11. TITLE *(Include Security Classification)* See box 19	

12. PERSONAL AUTHOR(S)
William M. Clifford, B.S., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ________ TO ________	1985 December 2	168

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
FIELD	GROUP	SUB. GR.	Artificial Intelligence, Robots, Robotics, Learning (NT Transfer of Training), Learning Machines
09	02		
05	10		

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

17 con't
06 04

Approved for public release: IAW AFR 190-1/.
[illegible]
LYNN E. WOLAVER 16 JAN 86
Dean for Research and Professional Development
Air Force Institute of Technology ~~(ATC)~~
Wright-Patterson AFB OH 45433

Title: Automating Knowledge Acquisition in a Flightline Robot.

Thesis Advisors: Steven Cross, Capt, AFIT/ENG, USAF
Timothy Anderson, AFAMRL, WPAFB OH

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED XX SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED

22a. NAME OF RESPONSIBLE INDIVIDUAL	22b. TELEPHONE NUMBER *(Include Area Code)*	22c. OFFICE SYMBOL
Steven Cross, Capt, USAF	513-255-3576	AFIT/ENG

DD FORM 1473, 83 APR   EDITION OF 1 JAN 73 IS OBSOLETE.   UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

Robots enjoy widespread use in industry on simplistic repetitive tasks in a controlled environment. Tasks in the military domain, particularly the flightline, require mobility and intelligence. While the mobility issue is being addressed, the intelligence issue is not. By giving the robot the ability to learn from a novice, the robot could be placed on the flightline and learn what it needs to know from the domain experts.

Rather than deal with a toy problem, this work takes the actual refueling "Job Guides" for two aircraft and shows how these can be used directly by a computer. The process involves three steps. First the text is transformed into a common natural language processor form. Second the forms are expanded by applying expert system and planning techniques to include missing domain and world knowledge. Finally the forms are examined to allow the commonalty between the two refueling tasks to be extracted.

Discussion includes background on learning, natural language processing using Conceptual Dependency representation, and planning using the Stanford Research Institute Problem Solver (STRIPS).